Ciphers By Ritter

Ritter Software Engineering

New Encryption Technologies for Communications Designers

Copyright © 1996-2006, Ritter Software Engineering.

Search Site: Go Go Google

Ciphers by Ritter Contents

GENERAL INFORMATION

- Controlling Patent Pornography (1997) Updated 2006-01-06
  - Patent Pornography, Patent Porn, patent porn, patent pornography, patent porn, patent porn, patent porn

Terry Ritter, P.E.

Hardware and Software Engineer, Chip System Architect *and* Cryptographic Engineer

Ritter Software Engineering
http://www.io.com/~ritter/
2609 Choctaw Trail
Austin, Texas 78745

Phone / Answer Machine / Fax: 512-892-0494
E-mail: ritter@io.com

Mr. Ritter has been researching, designing, implementing, measuring, and writing about cipher systems full time since 1989. *Ritter Software Engineering* is the business entity established to conduct and support this work.

"**As a Professional Engineer, I solve problems:** I look at the situation, identify the problem, research it, innovate a solution, apply it, check to see that it works, and then I move on. I design custom ciphers, and review system designs. I construct detailed written analysis of documents and arguments, and advocate positions as necessary." -- Terry Ritter, P.E.

Background *Lite*

PROFESSIONALLY

- **An independent professional since 1981**
- **Registered Professional Engineer**
- **Member: IEEE and ACM**

CURRENTLY

- **Independent Consulting Engineer** in cryptography
- **Supplier** of patented ciphering technology
- **Designer** of custom ciphers
- **Analyst** of existing systems and proposed designs

FORMERLY

- **Cryptography Consultant** to Bankers Trust
- **Software Designer** of dedicated multiprocessor LAN
  - Both closely-coupled and loosely-coupled real-time multiprocessing.
  - Fiber-optic main high-speed communications.
  - Full dynamic flow-control with error-check and retry.
- **Adjunct Assistant Professor of Electrical Engineering** at Georgia Tech
  - Wrote half the course and delivered half the lectures on developing software for microprocessor-based equipment designs.
  - Based the labs on early versions of BASIC09 and OS9.
- **Microprocessor Chip Architect** at Motorola
  - Co-architect and responsible for the specification of the MC6809 Advanced Microprocessor.
  - Responsible for the LS-TTL "breadboard" of the MC6809 NMOS design.
  - Responsible for the BASIC09 structured BASIC language project, with operating system OS9 and their associated specifications.
  - Also involved with MC6802, MC6845 CRTC, and MC6847 VDG, and other designs.
- **Fixed Ciphony Repair Specialist** in U.S. Army, Nha Trang Sig Bn (AUTODIN), South Vietnam, 1967-68.

AUTHOR of various works on cryptography, microprocessors, and other topics:

- **Ritter's Crypto Glossary *and* Dictionary of Technical Cryptography** (366K)
- **Learning About Cryptography** (51K)
- **1998 -- Usenet sci.crypt articles**
  - Orthogonal Latin Squares, Nonlinear Balanced Block Mixers, and Mixing Ciphers (31K)
  - Practical Latin Square Combiners (22K)
  - Break This 8-Bit Block Mixing Cipher (57K)
  - Break This 4-Bit Block Mixing Cipher (8K)
  - Measured Distant Avalanche in Large Block Ciphers (34K)
  - Measured Boolean Function Nonlinearity in Feistel Cipher Constructions (32K)
  - Measured Boolean Function Nonlinearity in Variable Size Block Ciphers (29K)
- **1998 -- JavaScript Articles**
  - Normal, Chi-Square and Kolmogorov-Smirnov Statistics Functions in JavaScript (50K)
  - Active Boolean Function Nonlinearity Measurement in JavaScript (41K)
- **1997 -- Usenet sci.crypt articles**
  - Measured Boolean Function Nonlinearity in Mixing Cipher Constructions (25K)
  - Chi-Square Bias in Runs-Up/Down RNG Tests (13K)
- **Crypto Articles in Print**
  - Estimating Population from Repetitions in Accumulated Random Samples (1994) (82K)
  - Voice and Video Cryptography in a DSP Environment (1991) (168K)
  - The Efficient Generation of Cryptographic Confusion Sequences (1991) (168K)
  - Transposition Cipher with Pseudo-Random Shuffling: The Dynamic Transposition Combiner(1991) (44K)
  - Substitution Cipher with Pseudo-Random Shuffling: The Dynamic Substitution Combiner (1990) (38K)
- **Other Topics in Print**
  - The Politics of Software Patents (1991) (30K)
  - The Great CRC Mystery (1986) (34K)
- **Older Microprocessor Articles**
  - Modular and Structured Programming on Small Systems (1980)
  - Varieties of Threaded Code for Language Implementation (1980)
  - A Microprocessor for the Revolution: The 6809 (1979)
  - Resident Memory Test Systems, With an Example for the 6800 (sic) (1978)
- **Older Directional Antenna and Electronics Articles**
  - The Absolute Field in Direction Antennas (1976)
  - The Directional Antenna Pattern (1975)
  - Controlling Op Amp Gain with One Potentiometer (1972)

INVENTOR of fundamental cryptographic components:

- **Dynamic Substitution** (U.S. Patent 4,979,832) (1990) -- A "cryptographic combiner," for invertibly yet nonlinearly mixing a stream-cipher keying sequence with data. Like exclusive-OR with strength.
- **Balanced Block Mixing** (U.S. Patent 5,623,549) (1997) -- A structure for invertibly mixing two input blocks into two resulting blocks in a statistically-balanced way. Some versions are nonlinear.
- **Variable Size Block Ciphers** (U.S. Patent 5,727,062) (1998) -- A structure for building block ciphers with dynamically-variable block size. When coupled with random-length header and trailer fields, it is hard to imagine how one might attack such a cipher, because one could not identify a block to work on.

DESIGNER of ciphers:

- **Fenced DES** (1994) -- "A larger and better DES." A 256-bit block cipher which uses DES as a component. Stronger than DES, and faster than Triple-DES.
- **Fencing and Mixing Ciphers** (1994) -- A variety of new block cipher designs using various forms of balanced mixing, as opposed to the usual Feistel "round-based" construction.
- **Variable Size Block Ciphers** (1995) -- A variety of new block cipher designs with dynamically-variable block size. These ciphers will avalanche the entire output block if even one bit of the input block changes, just as we expect from any true block cipher.

IMPLEMENTOR of software cryptosystems:

- **Dagger** (1993) -- A fast, flexible "drop in" 16-bit Dynamic Substitution cipher engine of moderate strength in 80x86 assembly language and portable C.
- **Penknife** (1993) -- A highly-developed Dynamic Substitution file cipher product for DOS, producing ASCII ciphertext for e-mail transmission
- **Cloak2** (1994) -- A highly-developed Dynamic Substitution file cipher product for DOS, with 992-bit secret keys.

PAST POET:

- Affected By Love (1972)

---

*Last updated:*1998-11-18

Update Log for *Ciphers By Ritter*

Changes to the Cryptography Pages

September 1998 through Present

Terry Ritter

1999-06-22

- New: **Experimental Characterization of Recorded Noise (1999) (11K index into 25 subpages with 3 .GIF graphs each)**. Characteristics of experimental noise generators are calculated and graphed.

1999-04-29

- New: **Fixing Strength Problems in Cipher Use (1999) (644K)**. A sci.crypt conversation on fundamental problems in cryptography.

1999-04-08

- Main Page Reorganization
- Update Log Reorganization

1999-03-30

- Major Update: **Random Noise Sources (25K + many .GIF's and .JPG's)**. Modern, simple, tested, battery-powered designs produce "unknowable" analog noise to be digitized by a sound card.

1999-03-09

- New: **Random Noise Sources**. Devices which produce "unknowable" analog noise.

1999-02-26

- Major Update: **Crypto Bookshop**.

1999-02-21

- Update: Usenet Discussions
  - New: Birthday Attack Calculations (57K)
  - New: Bit Counting and Similar Instructions (48K)
  - New: What is a "Group" in Block Cipher Analysis? (13K)
  - New: Block Cipher Modes for One-Block Messages? (17K)
  - New: Differential Cryptanalysis (7K)
  - New: Random Access to Encrypted Data (32K)
  - New: What's the Meaning of "Invert"? (37K)
  - New: The Hardware Random Number Generator (390K)
  - New: Patent Notebook Consequences (13K)
  - New: Software Patents? (278K)
  - New: The Pentium III RNG (100K)
  - New: Random Numbers in C (159K)
  - New: Random Numbers from a Sound Card (94K)
  - New: Software, Patents and Algorithms (213K)

1999-01-20

- Update: Usenet Discussions
  - New: The Value of Cryptanalysis (949K)

1999-01-19

- Update: Usenet Discussions
  - New: Junction Noise Experiments
  - New: Patents and Personal Use (68K)
  - New: Combiner-Type Algorithms
  - New: The Meaning of "Break"
  - New: The Several Types of Random
  - New: Tests for Randomness
  - New: The Homophonic Block Cipher Construction (213K)
  - New: AES and Patent Rights (306K)
- Update: **Crypto Glossary** (395K)
  - New: Ergodic
  - New: Stochastic
  - New: Stationary Process
  - New: Markov Process
  - Update: State
  - New: Process
  - Update: Mersenne Prime

1999-01-09

- Update: **Crypto Glossary** (392K)
  - Update: Finite Field
  - New: Galois Field
  - Update: GF 2^
  - Update: Primitive Polynomial
  - Update: Cryptography
  - Update: Code
  - Update: Cipher
- Update: **Learning About Cryptography** (66K)
  - Update: The Fundamental Idea
  - Update: The Most Important Book
- Major Update: **Crypto Bookshop**.
  - New book: **Cryptography and Network Security: Principles and Practice**, by William Stallings.
  - New book: **Discrete Mathematics Using Latin Squares**, by Charles F. Laywine and Gary L. Mullen.
  - And many others.

1998-12-29

- Update: **Crypto Glossary** (390K)
  - Update: Authentication
  - New: Dictionary Attack
  - Update: Secrecy
  - Update: Cryptography
  - New: Ciphony
  - New: Superencryption
  - New: Bel
  - New: Decibel
  - New: dB
  - Update: Power
  - New: RMS
  - New: Root Mean Square
  - New: Octave
  - New: Semigroup
  - New: Set
  - New: Closed
  - New: Associative
  - New: Commutative
  - New: Distributive
  - New: Fourier Theorem
  - New: Fourier Series
  - Update: FFT
  - Update: Fast Walsh-Hadamard Transform
  - New: Trust

1998-11-28

- Update: **Learning About Cryptography** (62K)
  - New: Naive Ciphers
  - New: Naive Challenges
  - Update: What Cryptography Can Do
  - New: System Design And Strength
  - New: Public Key Ciphers
- Update: **Crypto Glossary** (372K)
  - Update: Strength
  - Update: TEMPEST
  - New: Whitening
  - New: White Noise
  - New: Pink Noise
  - Update: Inductive Reasoning
  - Update: Statistics
  - New: Base-64
  - Update: ASCII

1998-11-18

- Update: **About the Author**.
- Update: **Crypto Glossary** (366K)
  - Update: Strength
  - Update: Random
  - New: Subjective
  - New: Objective
  - New: Absolute
  - New: Contextual

1998-11-15

- Update: **About the Author**.
- Update: **Crypto Glossary** (360K)

1998-11-14

- Update: **Crypto Glossary** (360K)
  - Update: Strength
  - Update: Mapping
  - New: Function
  - New: Inverse
  - New: Symmetric Group
  - Update: Block
  - New: MB
  - New: Mb
  - New: KB
  - New: Kb

1998-11-10

- Update: **Crypto Glossary** (355K)
  - Update: Attack
  - Update: Multiple Encryption
  - New: Birthday Paradox
  - New: Birthday Attack

1998-10-25

- Update: **Crypto Glossary** (350K)
  - New: Mixing Cipher
  - New: Inductive Reasoning
  - New: Deductive Reasoning
  - New: Fallacy

1998-10-22

- Update: **Ritter's Net Links** (40K)

1998-10-21

- Update: **Crypto Glossary** (343K)
  - Update: Transistor
  - New: Monadic
  - New: Dyadic
  - New: Unary
  - Update: Binary
  - New: Octal
  - New: Decimal
  - New: Hexadecimal
  - Update: Break
  - New: A Cipher Taxonomy
  - Update: Cryptanalysis
  - Update: Attack

1998-09-22

- New: **Orthogonal Latin Squares, Nonlinear Balanced Block Mixers, and Mixing Ciphers** (31K)
- Update: **Practical Latin Square Combiners** (22K)

1998-09-17

- New: **Practical Latin Square Combiners** (22K)

1998-09-04

- Update: **Learning About Cryptography** (51K)

To Earlier Log (Sept 1995 - Aug 1998)

Ritter's Net Links

Cryptography and Net Resources (Including Major Crypto Links Sites)
A *Ciphers By Ritter* Page

Contents

- **News** -- World, Technical, Crypto
- **Searching** -- Meta Search, Usenet News, Web Page, White Pages
- **On-Line Sales** -- Computers, Other
- **Crypto (General)** -- People, Resources, Lists of Links, Classical
- **Crypto (Specific)** -- Ciphers, Designs, Products, Claims, Misc.
- **Patent Links** -- PTO, General, Law, Searching
- **Randomness Links** -- This Site, General, Pseudorandom, Really Random, Tests
- **Austin Links**
- **Misc** -- Archives, Organizations, Molecular Computing, US Government, Etc.

Net Links

World News

- ABCNEWS.com
- CNN News
- FOX News
- HotBot
- Infoseek
- WebCrawler
- Yahoo
- News Index
- Newshub -- Summary of today's newest news, updated every 15 minutes, every day!
- TotalNEWS: all the News, on the Net, all the Time
- The Nando Times
- Rain or Shine Weather
- Weather: Welcome to AccuWeather®
- NPR-- National Public Radio Online

Technical News

- Welcome to NEWS.COM
- ZDNNews: Page One
- TechWeb News
- Newsbytes
- Wired News
- EET-i -- Electronic Engineering Times Interactive
- Electronic News On_Line
- Welcome to EDN Access
- Computer Currents Interactive
- Welcome to PC Week
- PC MAGAZINE on the WEB

Crypto News

- Cryptome
- CyberTimes Articles on Encryption
  - See 1997-10-09 Article: *Europeans See U.S. Encryption Proposal as Threat to Privacy* By Edmund L. Andrews
- The Center for Democracy and Technology
- Electronic Privacy Information Center
- Encryption Policy Resource Page
- ACLU: American Civil Liberties Union
- Senator Burns' Encryption page
- PCCIP Web Main Menu
- OECD Online
  - Security, privacy, cryptography and intellectual property rights
  - Cryptography Policy Guidelines
  - Untitled Document
- Welcome to Europa
  - European Commission
  - Information Society Project Office
    - European Internet Forum Policy Papers
    - COM(97) 503
- Americans for Computer Privacy
- Advanced Encryption Standard (AES) Development Effort
- Eavesdropping - Codex Surveillance & Privacy Page - Surveillance

Searching

- **Meta Searching**
  - MetaCrawler
  - Inference Find! -- Home Page and Resource Center
  - Ask Jeeves
  - ProFusion
  - SavvySearch
  - all 4 one
  - Highway 61
  - Dogpile
- **Usenet News Searching**
  - DejaNews
    - Deja News - Browse sci.crypt
    - Deja News - Search for sci.crypt
  - Reference.COM
    - Reference.COM Advanced Search sci.crypt
  - Alta Vista
  - HotBot
  - Excite
  - InfoSeek
- **Web Page Searching**
  - Alta Vista
  - Yahoo
  - HotBot
  - Northern Light Search
  - InfoSeek
  - Excite
  - WebCrawler
  - Lycos
  - What-U-Seek
  - PlanetSearch - Your Web. Your Way.
- **White Pages Searching**
  - Four11
  - Bigfoot
  - WhoWhere?
  - internet @ddress.finder
  - GTE SuperPages
  - Phone & Address - InfoSpace

On-Line Sales

- **Computers**
  - MidWest Micro Home Page
  - Gateway
  - CDW - Computer Discount Warehouse
  - Insight
  - NECX - Home & Office Computer Center
  - Micro Warehouse Inc. : Your #1 Source for All your Computer Needs Online!
- **Other**
  - Ronco, Inc.

Crypto People

- People's home pages (links to crypto people)
- Cryptographers Homepages and publication pages
- Links for Cryptographers and Computer Security People
- ETH Zurich - Computer Science - Jan Camenisch -- Cryptographers
- Cryptographers
- Eli Biham's Home Page
- Serge Vaudenay
- S. E. Tavares
- Luke O'Connor

Crypto Resources

- http://www.jya.com/crypto.htm -- amazing crypto document archive
- http://www.itd.nrl.navy.mil/ITD/5540/ieee/cipher/ -- Cipher - Newsletter of the IEEE CS TC on Security and Privacy (HTML)
- http://www.swcp.com/~iacr/ -- International Association for Cryptologic Research
- ftp://ftp.psy.uq.oz.au/pub/Crypto/ -- DES, SSL
- ftp://rpub.cl.msu.edu/pub/crypt/sci.crypt -- sci.crypt Archives
- ftp://ripem.msu.edu/pub/crypt/sci.crypt/ -- sci.crypt Archives
- http://www.nsa.gov:8080/museum/tour.html -- Museum Tour
- http://www.aegeanpress.com/books/ -- Aegean Park Press
- North American Cryptographic Software Access
- Beginners' Guide to Cryptography
- Stream Cipher Review.
- Welcome to Signal 9 Solutions: Virtual Private Network - VPN - and Internet Security
- Cryptography -- CryptoTool
- Cryptography and Encryption: Miscellaneous Topics
- Craig Raskin's Cryptography Home Page
- Classical Cryptography
- Workshop on Selected Areas in Cryptography (SAC)
- -|-|-|-|-|-|- No More Secrets! -|-|-|-|-|-|-|-
- Cryptography for Information Security
- Fortify for Netscape - Home Page
- Pointers to Cryptographic Software
- David Hopwood - Cryptix
- Ergodic Theory
- The Miraculous Bailey-Borwein-Plouffe Pi Algorithm
- Pi and Other Constants

Lists of Crypto Links

- Security and Encryption Links (by Peter Gutmann)
- Ronald L. Rivest : Cryptography and Security
- bsy's Security Related Net-pointers
- Pat's Crypto Sources Hotlist
- Cryptography Resources
- Cryptography Resources
- Cryptography links
- Richard Pinch: Cryptography links
- Crypto-Log: Guide to Cryptography Resources (a massive but dated site)
- ShowDowns Encryption Site
- Cryptography and Encryption: Miscellaneous Topics
- CRYPTO_LINKS
- Cryptographic Links
- Cryptography Links Outside of North America
- North American Cryptographic Software Access
- Security and Cryptography
- Ian Nelson's Cryptography page
- Computer security and cryptography
- Tom Dunigan's Security page
- Cryptography URL
- Interesting Web Sites
- Lewis' Cryptography and Computer Security Links
- Documents about cryptography
- Agents
- Sean's obsessions..
- Interesting Web Sites
- Interesting Web Sites
- Cyber Data - web page design / web page designers / developers / authoring - Adelaide, Australia - Encryption and Security
- Quadralay Cryptography Archive

Classical Crypto

- John Savard's Home Page
- Classical Cryptography
- CipherClerk Introduction
- The Crypto Drop Box
- Enchanted Mind - Creative Cryptography
- Beginners' Guide to Cryptography

Ciphers

- Cryptographic Algorithms
- An Introduction to Cryptography
- Some Relevant Aspects of Cryptography: An Overview.
- Cryptographic Algorithms
- CipherClerk's List Of Ciphers
- Cryptographic Libraries: A comparison
- Cryptography Section
- Cryptography

Crypto Designs

- Mercy: a fast cipher for disk block encryption
- Enkryptonator Home Page
- John Savard's Home Page
- TecApro Internacional
- ClassicSys, a new way to move on cryptography
- WTShaw Web Site
- Method and Apparatus for the Encryption, Decryption and Authentication of Messages Using Dynamical Systems
- Howard Gutowitz's Home Page

Crypto Products

- Cryptography -- CryptoTool
- Enkryptonator Home Page
- Baltimore Technologies : J/CRYPTO v2
- DataTech Systems - Home Page
- Kremlin Encryption Suite -- easy to use and secure encryption and file deletion
- ENCRYPTION
- Simulation Laboratories, Inc.
- Welcome to Wave Systems Corp.
- Invincible Data Systems, Inc.: PGP (or Pretty Good Privacy) - compatible encryption software for e-mail security, hard disk encryption utilities for stored data protection, and access control hardware tokens
- Jetico Home page
- Utimaco Home
- Eutron's Home page
- Speak Freely for Windows
- Mykotronx, Inc. Product Overview
- Rainbow Technologies Home Page
- computer security - Innovative Security Products
- Genio USA, Homepage
- DELTA COMMUNICATIONS -- Data Communications Specialists -- high speed data communications
- Cisco Connection Online by Cisco Systems, Inc.
- Network Security
- DATA PRIVACY FACILITY (DPF)
- KarlNet's KarlBridge and KarlRouter
- Remote KarlBridge
- SLI Model 700 CSU/DSU
- Downloads Security Software
- Cryptography products listing
- Certicom Corp. Cryptographic Technologies
- AR - Algorithmic Research Data Security Solutions
- RSA Data Security, Inc.
- Cylink Corporation
- Internet Secure
- SynCrypt: Encryption, Digital Signature, Public Key Cryptography, and Steganography Security Software by SynData
- Security Dynamics | Home Page

Crypto Claims

- Welcome to Signal 9 Solutions: Virtual Private Network - VPN - and Internet Security
- gamma publishing is pleased to announce the publication of:
- Ultimate Privacy Corporate Profile
- Provably Secure and Efficient Block Ciphers

Crypto Misc

- The Australian Cryptography FAQ
- The Block Cipher Lounge
- The DES Analytic Crack FAQ
- The Laymen's Guide to the Entrails of RSA
- Implementing Elliptic Curve Cryptography
- S/MIME Freeware Library
- S/MIME Working Group
- CSC Info
- Encryption
- Enkryptonator Editorial Page
- Mach5 Software Cryptography Archives
- Adam Shostack's Homepage
- Experimental reference center for cryptographic data protection
- Latin Squares
- Engineering and physics
- http://www.plu.edu/~spillmrj/cap.html
- The Complete, Unofficial TEMPEST Information Page
- Electromagnetic Pulse (EMP) and TEMPEST Protection for Facilities
- Directory of /pub/papers

Patent Links

- **The USPTO**
  - United States Patent and Trademark Office
  - Manual of Patent Examining Procedure, 6.2
- **General Information**
  - General information about patents
  - Means Plus Function Patent Claims
  - The Trudel Group
  - Can You Really Get a Patent Without a Lawyer?
- **Patent Law and Rules**
  - Patent Law Materials
  - 35 USC CHAPTER 10 (01/24/94)
  - Patent Laws and Rules
  - Patent Law
  - Substantive Patent Law
  - Patent Portal: Internet Patent Resources
  - Jones & Askew - Patent Law References
- **Patent Searching**
  - COS Patent Citation Database
  - STO's Internet Patent Search System
  - IBM Patent Server

Randomness Links

- **This Site**
  - various Literature Surveys (specifically linked below)
  - The Efficient Generation of Cryptographic Confusion Sequences (1991) (168K)
- **General**
  - RNG Surveys and Related Topics (16K)
  - G J Chaitin Home Page
  - CME's Random Number Conditioning Page
  - News on Random Number Generators
  - Entropy on the World Wide Web
  - Lane A. Hemaspaandra's Home Page
- **Computer: Pseudorandom**
  - RNG Implementations and Related Topics (26K)
  - A Practical "Perfect" Pseudo-Random Number Generator
  - Random Number Generators (RNGs)
  - Random Number Generator Software Catalog
  - Randomness resources for Dr. Dobb's Journal readers
  - Random number generators -- The pLab Project Home Page
  - The World-Wide Web Virtual Library: Random Numbers and Monte Carlo methods
  - Pseudorandom Generators
  - Random Number Generation, Taygeta Scientific Inc.
  - Mersenne Twister: A random number generator
  - Random Number Generators
- **Physical Measurement: Really Random**
  - Random Noise Generator
  - Ph 77 - The Geiger Counter And Counting Statistics - September 1997
  - Random Electrical Noise (34K)
  - Random Number Machines (62K)
  - Random number generators
  - RBG1210
  - ORION RNG - Home
  - ORION RNG - Home
  - The ComScire QNG From Quantum World
  - SG100
  - Aware Electronics Corp.
  - Welcome to lavarand!
  - HotBits: Genuine Random Numbers
  - Hardware Random Bit Generator
  - CME's Random Number Conditioning Page
  - Noisemaker schematic
- **Tests**
  - Randomness Tests and Related Topics (39K)
  - Tests for Random Numbers
  - Pseudorandom Number Sequence Test Program
  - Random Number Results
  - ORION - RNG LISTING
  - DIEHARD

Austin Links

- Austin City Links
- Austin Business
- Austin 360: THE city site for Austin.
- Austin Software Council
- Austin360: Weather: KVUE-24 Weather Top
- FOX 7
- Office Cam
- FOX 7 -- Live Images
- itmWEB: Austin, Texas, USA WebCam

Misc

- **Archives**
  - oak software repository
  - Washington University Data Archive
  - usenet-by-hierarchy -- FAQ's
  - RFC's
- **Organizations**
  - IEEE Home Page
  - ACM, The First Society in Computing
- **Molecular Computing**
  - http://www.cs.princeton.edu/~dabo/biocomp.html -- Publications on Molecular Computers
  - Don Beaver
  - http://dope.caltech.edu/winfree/DNA.html -- Erik's Molecular Computation page
  - http://www.cs.princeton.edu/~dabo/bio-comp/molecular.html -- WIRED 3.08: - Gene Genie By Thomas A. Bass
- **US Government**
  - THOMAS: Legislative Information on the Internet
  - Library of Congress World Wide Web Home Page
  - U.S. Postal Service
  - GPO Access on the Web
  - FedWorld Information Network Home Page
- **Atlantis**
  - ATLANTIS THE FINAL SOLUTION? - AN INTERVIEW WITH ALAN F. ALFORD
  - Viatcheslav Koudriavtsev's Hypothesis on Plato's Atlantis
  - Ancient Civilizations - Atlantos. Lemuria, Mu
  - Atlantis
  - A.R.E.- Ancient Civilizations
  - Atlantis - The Lost Continent Finally Found
- **CRC**
  - Cyclic Redundancy Check (CRC) C Source Code
  - Galois Field (Florent Chabaud)
- **Latin Squares**
  - JavaScript - Experimental Design
  - Table of contents of Volume 2 of the electronic journal of combinatorics
  - Latin Rectangle
  - Generating uniformly distributed random latin squares
- **Etc.**
  - Decompilation of Binary Programs - dcc
  - Where is the archive for newsgroup X?
  - ftp://ftp.netcom.com/pub/dj/djames/lynx/catalogs.html -- Library Catalogs
  - http://www-dsed.llnl.gov/documents/WWWtest.html -- WWW Viewer Test Page
  - Web Publishers - Search Engines Front Page
  - whatis.com
  - VLSI Cores, Inc.
  - Welcome to Browser Wars
  - Net Talk Live! The Internet Talk Show
  - Exploratorium: ExploraNet
  - Expedia - Full Circle
  - TheTrip.com: theFLIGHT
  - Found Money Homepage
  - Mr. Showbiz Movie Guide
  - Information for Consumers
  - itmWEB: Information Technology & Systems Management
  - "Find A Friend" - Social Security Number Search - Tracing - Search Information - How to find people
  - Optimizing assembly code
  - iVillage: The Women's Network & Chats, women, parenting, health, sex, career, jobs, employment, work from home, relationships, fitness, beauty,food, cooking, money, investing, children, sports, home, recipes, jobs, love, employment
  - Virtual London - London-Cam Page - Suivez l'évolution du temps sur Londres
  - FIP CAM!
  - Opera Software - Bringing speed and fun back into Internet browsing
  - ZDNet Webopædia
  - Time Warner's Pathfinder!
  - My Virtual Reference Desk - A One-Stop Site for All Things Internet
  - Daily Usenet report for hiram.io.com: index
  - 4001 Duval Hair Salon
  - Timely Technical Topics
  - Ciphers By Ritter

Terry Ritter, his *current address*, and his *top page*.

Last updated: 1998-10-22

whatUseek intraSearch

sign up
learn more
login

whatUseek intraSearch
get your own site-level search engine

- Add a professional-level search engine to your web site in five minutes or less.
- whatUseek intraSearch site search engine is FREE!
- Know who your web site visitors are and what they're looking for with on-the-fly reports.
- Customize the look and feel of your site's search engine.

Building a Personal Portal is easy!

- Create an intraSearch account and then login to your account (NOTE: For each Personal Portal you want to create, create a separate intraSearch account)
- In your Account Settings page, go to Account Settings >>Web Site Information >>Site Name and URLs
- Enter the URLs of the sites you want to add to your Personal Portal in the Secondary Entrypoints box. Press the Update button when you're done.
- Go to Spidering Options >> Re-Crawl Your Site/Update Search Index >> Re-Index Now

Your Personal Portal is complete, and ready to be searched as soon as our spiders crawl the sites you've specified!

whatUseek's intraSearch powers the site search engines for over 120,000 Web sites.

Set up your own site search engine / personal portal in less than five minutes.

EXAMPLE: Separate family web sites are all linked together through a common family search engine using intraSearch.

EXAMPLE: A Palm computing enthusiast creates his own Palm portal using intraSearch.

Bridging the Last Mile of Search! ™

WEBMASTER RESOURCES

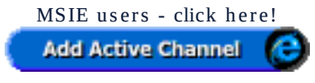
whatUseek intraSearch add url web site hosting web site promotion
whatUseek HyperSuite your own email service domain registration
code search meta search free email

THE WHATUSEEK NETWORK

home add url free file storage advertise
contact whatUseek intraSearch chubba KingCode

Copyright © 1998-2000
Legal Information
Privacy Policies

MSIE users - click here!



Learning About Cryptography

A Basic Introduction to Crypto
A *Ciphers By Ritter* Page

Terry Ritter

Current Version: 1999 Jan 09

For some reason, good cryptography is just much harder than it looks. This field seems to have a continuous flow of experts from other fields who offer cryptographic variations of ideas which are common in their other field. Now, there is nothing wrong with new ideas; but there is not much more reason to expect that a new idea from a mathematician will be good in cryptography than would be the case if the idea came from a chemist or a farmer. It is not that these "outsiders" ideas have been previously tried and found wanting; it is just that we have no reason to expect that they will be good.

You may wish to help support this work by patronizing Ritter's Crypto Bookshop.

Contents

- The Fundamental Idea of Cryptography
- A Concrete Example
  - A Simple Cipher
  - Encipering
  - Deciphering
  - Simple Transformations
  - More Transformations
  - Weak or Strong Transformations
  - Keyspace
  - Digital Dictionary Ciphering
  - Huge Keys
- Block Ciphers
- Stream Ciphers
- What Cryptography Can Do
- What Cryptography Can Not Do
- Cryptographic with Keys
- Problems with Keys
- Cryptography without Keys
- Strength
- Cryptanalysis
- Secret Keys and Strength
- Cryptanalysis versus Interdiction
- Secret Ciphers
- Hardware vs. Software Ciphers
- Block Ciphers
- Stream Ciphers
- Public Key Ciphers
- The More Important Book
- Classical Cryptanalysis
- Other Books
- Coding Theory
- Cryptanalysis
- For Designers

The Fundamental Idea of Cryptography

It is possible to transform an *intelligible* message or plaintext into an "unintelligible" form or *ciphertext* in which the information is present but hidden. This can not release the transformed message (the ciphertext) to anyone who can do the inverse transformation.

By using different transformations, we can create many different ciphertexts for the same basic message. Yet if we select a particular transformation, or "*cipher*," we can hope that anyone wishing to recover the original message or "*plaintext*" can do so better than simply trying all similar transformations until one of the results looks like a reasonable message.

The difference between intermediate forms is the interpretation of the ciphertext. Different ciphers and different keys will produce different interpretations (*different plaintexts*) for the exact same ciphertext. Thus, we can choose an appropriate plaintext by choosing the appropriate cipher or key.

Naturally, the intended recipient needs to know how to transform or *decipher* the intermediate form back into original message, and this is the key distribution problem.

In fact, cryptanalysis is literally meaningless, in the sense of having no one clear interpretation. In so-called *perfect* ciphers, and ciphertext of appropriate size can be interpreted as any message, just by selecting an appropriate key. In fact, any number of different messages can produce exactly the same ciphertext, by using the appropriate keys. In other ciphers, this may not always be possible, but it may always be misleading. Thus, if we do not know the right key, we can not be certain that the message we guess from ciphertext is the exact plaintext message which was sent.

A Concrete Example

Most of our focus encountered a simple form of ciphering in grade school, and it usually goes something like this:

A Simple Cipher

On a piece of lined paper, write the alphabet in order, one character per line:

```
a
b
c
d
e
f
g
h
i
j
k
l
m
n
o
p
q
r
s
t
u
v
w
x
y
z
```

Then, on each line, we write another character to the right. In this second column, we also want to use each alphabetic character exactly once, but we want to place them in some different order.

```
a   x
b   y
c   z
d   a
e   b
f   c
g   d
h   e
i   f
j   g
k   h
l   i
m   j
n   k
o   l
p   m
q   n
r   o
s   p
t   q
u   r
v   s
w   t
x   u
y   v
z   w
```

When we have done this, we can take any message and encipher it letter by letter.

Enciphering

To encipher a letter, we find that letter in the left column, then use the associated letter from the right column and write that down. Each letter in the right column then becomes a substitute for the original letter.

Deciphering

Deciphering is similar, except that we find the ciphertext letter in the right column, then use the associated plaintext letter from the left column. This is a little harder, because the letters in the right column are not in order. But if we attempt to do this manually, we could make a lot where the ciphertext letters were in order; this would be the reverse of the enciphering transformation. And if we have both lists, deciphering is just as easy as enciphering.

The Simple Transformation

This simple cipher is an example of a simple substitution cipher, a monoalphabetic substitution cipher, or a simple substitution. The "transformation" is the particular arrangement of letters in the second column, a permutation of the alphabet. There can be many such arrangements. But in this case the key is that particular arrangement. We can copy it and give it to someone and then send enciphered messages. We can change it—or, if we suspect that someone has a copy, we can not use it any more. We can also use it to keep a secret between ourselves.

Simple Transformations

Our suppose we have a full notebook of lined paper, each of which contains a different arrangement in the second column. Suppose each page is numbered. Now we can pick a number and encipher each plaintext letter by using the page with that number. Suppose we have 100 such pages, which is a set of 100 different ciphers. We can call this set the transformation. In order to encipher a message we must give where to find the proper page for each letter of the transformation in the book before finding the right one. Since enciphering the notebook does not immediately expose our message, maybe we can use the same transformation for a number of letters before we change to another page. Maybe we can use the same notebook for the next few messages to each friend. Different people can use the same notebook and yet still cipher messages that are not readable by others.

Note that there is some potential for confusion in first ciphering the notebook or *transformation* a line, and then ciphering the number which selects that transformation also a line. Both of these are so similar, that we must be careful to keep them completely separate. We can not use the same line for both purposes.

Weak and Strong Transformations

One of the disappointments in a simple pencil cipher is very weak, because it "leaks" information. The more often a particular plaintext letter is used, the more often the associated ciphertext letter appears. And since language uses some letters more than others, simply by counting the number of times each ciphertext letter occurs we can make a good guess about which plaintext letter is represented. This is a weakness which we would like to avoid, if we could do so. Unfortunately, it is not clear that we can avoid it. We can try to make it difficult, by using the big list of lines, but we still have the underlying transformation from one language to another.

A "weak" cipher will have a few simple transformations. For example, the usual 26 line Caesar cipher with encipher 9 different letters at the same time, while the remaining 17 will encipher only through 8 different ciphertext letters. This is still pretty easy to cryptanalyze, even with a large alphabet. Instead of using 26 lines, a 64-line block cipher cross such as 2nd-order Beaufort takes us to a keyspace letter, which is something like 1026,0026,0026,0026,0026,0026,0026,0026.

Digital Dictionary Ciphering

One of the disappointments in a simple pencil alternative machine ciphering is that it operates very, very fast. This means that someone can try a very large number of plaintext keys, and yet still remain a "reasonable" keyspace exponent, about 2^27 keys, which sounds like a lot. Unfortunately, special purpose hardware could try this many deciphering in under 5 seconds. Thus, we need a much stronger cipher than a simple pencil cipher.

Different keys are used to select the proper page for each letter of the message. Thus, we can use a very large number of different ciphers. Suppose we have 100 pages, each with 26 lines, and we use 9 lines to select the proper page for each letter of the message. Then we have 100^9 or 1,000,000,000,000,000,000 different ciphers. This is a big keyspace, 59.7 bits, which is the amount of storage usually consumed by 10 six-character, a very minimal amount. A 128 bit key is "strong enough" to defeat even astronomically-powered attacks.

Huge Keys

Under the theory that if a cipher is good, a lot better, some people suggest using huge keys of 50,000 bits, or 1,000,000 bits, or even more. We can build such devices, and they can operate quickly. We can encipher 1 million bits per second, for example. What we do not have is a reason to believe that such a cipher is "strong." Instead of a cipher that is vulnerable to attacks from those with the right key, we might now have a cipher that is vulnerable, even immense large cannot provide never enough "strong." And while different attacks may show that the cipher actually has less strength, a huge keyspace does not guarantee that there are not such attacks.

Suppose we have a cipher with a 1 million bit key, and we want to be sure that it is "strong." Suppose, if we can, that we know of a way to reduce the cipher to a very special form, in which there are only two values in any cryptogram and, in which, we know that the actual keyspace is much smaller than the size of the key would indicate. For this reason, public key systems need to be 1,000 times larger than a 128 bit key, while delivering strength comparable to 128 bit secret key ciphers.

Naive Ciphers

Suppose we want to hide a name. We might think to innocent a different size for each letter. We might say: "First we have 7", but 1 is the 3rd letter in hunkle' so we write 3." We can continue this way, using the number of letters in each word to hide the next letter of our message.

This is a cipher, but it is not a secure cipher for decipherable. Anyone who knows the rules can decipher our message. Unfortunately, this is not of course not the intended goal of the cipherer(s) to decipher.

We have a problem in transmitting the rules, since the rules constitute the key. If we choose from among many rules, to be patient at all, we may have a strong cipher, but he unable to remember the key. And if we write the key down, all someone has to do is find that simple interpretive (which may be another encryption) key. So we might choose among the rules, in some pattern, which we can remember, but which is not obvious to others.

Another problem is that we are supposed to allow us to be long-winded. This sort of scheme tends to be very verbose, or perhaps just more long. For a longer message, which is more verbose? Maybe we can hide a lot of information in the fact that we have to be long-winded, and still be very, very, verbose.

Suppose we have 256 different rules, and we want to be able to encipher any single character, one character at a time, one character per message. Tend to send the message to someone who, we must somehow transport the rule to decipher the message. We can use the 256 rules to encipher the 256 possible characters, so sender why we cannot assume we can transport the rules?

Modern ciphering is about communications which attempt to solve these problems. A modern cipher has a large keyspace, which might well be controlled by a hashing computation on a language phrase or passwords. Instead of transmitting the cipher, we can transmit the password, and the receiver can use it to generate the cipher. Thus, we need only to transport a very small key, and yet we can use a very large number of different ciphers.

Moreover, in a modern cipher, we expect the key to be required, even if the Opponents know both the plaintext and the associated ciphertext for many messages to known-plaintext attack to be, not normally assume the Opponents know the full construction of the cipher, and but lots of known plaintext, and still cannot find the key. Such designs are we mean.

One possible application for cryptography is to secure on-line communications between work and home, perhaps leading to a society-wide reduction in driving, something we could all appreciate.

What Cryptography Can Not Do

Cryptography can only hide information which is encrypted and which it remains encrypted. But secret information generally does not stay encrypted, so there is normally an original period during which the secret is not protected. And secret information generally is not used in encrypted form, so it is again outside the cryptographic envelope every time the secret is used.

Secrets are often created in plain form, and subsequently encrypted or transmitted in the secret code which that secret is.

And while cryptography can hide secrets, it cannot hide:

- Physical attempts
- Gifts
- Careless meetings and training,
- Social engineering and bribery,
- Not-enough-literate wife on mistake means of support, or
- Actions.

And cryptography simply cannot protect against:

- Keyboard loggers,
- Underscreen spying,
- Bugs,
- Photographic evidence, or
- Extortion.

It is a little-to-imagine that cryptography alone could protect most information against Government investigations. Cryptography is only a small part of the protection needed for "absolute" secrecy.

Cryptography with Keys

Usually, we arrange to select among a large number of possible intermediate forms by using some sort of "pass phrase" or key. Normally, this is some moderately-long language phrase which we can remember, instead of something hard to write down (and hard to remember and hard to keep secret).

Those who have use of the original keys can expose the information hidden in the message. This reduces the problem of protecting information to:

1. Protecting the key, and
2. Protecting data.

This is similar to locking our possessions in our house and keeping the keys in our pocket.

Problems with Keys

Pass phrase is a small minority use of various things that can go wrong with keys:

- We can lose our keys.
- We can forget our keys.
- We can give a key to the wrong person.
- We can be robbed.
- Somebody can pick the lock.
- Somebody can go through a window.
- Somebody can bribe a locksmith.
- Somebody can slip a master key, and, physically for his.
- Somebody can get a warrant, then legally do whatever is required.
- Somebody can hide the house, then making everything worthless.

Even absolutely perfect keys cannot solve all problems, not can they guarantee privacy. Indeed, when cryptography is used for communications, generally at least two people know what is being communicated. Some of the problems with keys are:

- Someone may be coerced.
- To someone else.
- Through deceptive misunderstanding.
- So strange, for actions not or imagined.
- Under torture.
- Under duress.
- Or testimony.

When it is substituting keys could be an extensive set

A *Cipher's By Ritter* Page

Buy this book from amazon.com
Buy this book from Amazon.co.uk
Buy this book from Amazon.de
Buy this book from Amazon.fr
Buy this book from Amazon.co.jp
Buy this book from Amazon.ca
Buy this book from Amazon.it
Buy this book from Amazon.es
Buy this book from Amazon.com.br
Buy this book from Amazon.com.mx
Buy this book from Amazon.com.au
Buy this book from Amazon.com.sg
Buy this book from Amazon.com.tr
Buy this book from Amazon.com.cn
Buy this book from Amazon.com.in
Buy this book from Amazon.com.ca
Buy this book from Amazon.com.ar
Buy this book from Amazon.com.pl
Buy this book from Amazon.com.nl
Buy this book from Amazon.com.se
Buy this book from Amazon.com.be
Buy this book from Amazon.com.ae
Buy this book from Amazon.com.sa
Buy this book from Amazon.com.eg
Buy this book from Amazon.com.ng
Buy this book from Amazon.com.pk
Buy this book from Amazon.com.bd
Buy this book from Amazon.com.kh
Buy this book from Amazon.com.np
Buy this book from Amazon.com.lk
Buy this book from Amazon.com.mm
Buy this book from Amazon.com.af
Buy this book from Amazon.com.kw
Buy this book from Amazon.com.bh
Buy this book from Amazon.com.om
Buy this book from Amazon.com.ph
Buy this book from Amazon.com.co
Buy this book from Amazon.com.pe
Buy this book from Amazon.com.ve
Buy this book from Amazon.com.ec
Buy this book from Amazon.com.uy
Buy this book from Amazon.com.gt
Buy this book from Amazon.com.do
Buy this book from Amazon.com.sv
Buy this book from Amazon.com.pr
Buy this book from Amazon.com.jo
Buy this book from Amazon.com.ly
Buy this book from Amazon.com.ma
Buy this book from Amazon.com.dz
Buy this book from Amazon.com.tn
Buy this book from Amazon.com.ye
Buy this book from Amazon.com.sy
Buy this book from Amazon.com.iq
Buy this book from Amazon.com.lb
Buy this book from Amazon.com.ge
Buy this book from Amazon.com.az
Buy this book from Amazon.com.kz
Buy this book from Amazon.com.ua
Buy this book from Amazon.com.ru

Ritter's Crypto Glossary *and* Dictionary of Technical Cryptography

Technical Cryptographic Terms Explained

Hyperlinked definitions and discussions of many cryptographic, mathematic, logic, statistics, and electronics terms used in cipher construction and analysis.

A *Ciphers By Ritter* Page

Terry Ritter

Current Edition: 1999 Jan 19

For a basic introduction to cryptography, see Learning About Cryptography. Please feel free to send comments and suggestions for improvement to: ritter@io.com. You may wish to help support this work by patronizing Ritter's Crypto Bookshop.

Contents

A

Absolute, AC, Additive Combiner, Additive RNG, Affine, Affine Boolean Function, Alphabet, Alternative Hypothesis, Amplifier, Amplitude, Analog, AND, ASCII, Associative, Asymmetric Cipher, Attack, Augmented Repetitions, Authentication, Authenticating Block Cipher, Autokey, Avalanche, Avalanche Effect

B

Back Door, Balance, Balanced Block Mixer, Balanced Block Mixing, Balanced Combiner, Base-64, Bel, Bent Function, Bernoulli Trials, Bijective, Binary, Binomial Distribution, Birthday Attack, Birthday Paradox, Bit, Block, Block Cipher, Block Size, Boolean, Boolean Function, Boolean Function Nonlinearity, Boolean Logic, Boolean Mapping, Break, Brute Force Attack, Bug, Byte

C

Capacitor, CBC, c.d.f., CFB, Chain, Chaos, Chi-Square, Cipher, Cipher Taxonomy, Ciphering, Ciphertext, Ciphertext Expansion, Ciphony, Circuit, Clock, Closed, Code, Codebook, Codebook Attack, Combination, Combinatoric, Combiner, Commutative, Complete, Component, Computer, Conductor, Confusion, Confusion Sequence, Congruence, Contextual, Conventional Cipher, Convolution, Correlation, Correlation Coefficient, CRC, Cryptanalysis, Cryptanalyst, Cryptographer, Cryptographic Mechanism, Cryptography, Cryptography War, Cryptology, Current

D

dB, DC, Debug, Decipher, Decryption, Deductive Reasoning, Defined Plaintext Attack, Degrees of Freedom, DES, Decibel, Decimal, Design Strength, Deterministic, Dictionary Attack, Differential Cryptanalysis, Diffusion, Digital, Diode, Distribution, Distributive, Divide and Conquer, Domain, Dyadic, Dynamic Keying, Dynamic Substitution Combiner, Dynamic Transposition

E

ECB, Electric Field, Electromagnetic Field, Electronic, Encipher, Encryption, Entropy, Ergodic, Extractor, Exclusive-OR

F

Factorial, Fallacy, Fast Walsh Transform, FCSR, Feistel Construction, Fenced DES, Fencing, Fencing Layer, FFT, Field, Finite Field, Flip-Flop, Fourier Series, Fourier Theorem, Fourier Transform, Frequency, Function, FWT

G

Gain, Galois Field, Gate, GF 2^n, Goodness of Fit, Group

H

Hamming Distance, Hardware, Hash, Hexadecimal (Hex), Homophonic, Homophonic Substitution

I

IDEA, Ideal Secrecy, i.i.d., Inductive Reasoning, Inductor, Injective, Insulator, Integer, Intermediate Block, Interval, Into, Inverse, Invertible, Involution, Irreducible, IV

J

Jitterizer

K

KB, Kb, Kerckhoff's Requirements, Key, Key Distribution Problem, Keyspace, Keyed Substitution, Known Plaintext Attack, Kolmogorov-Smirnov

L

Latency, Latin Square, Latin Square Combiner, Layer, LFSR, Linear, Linear Complexity, Linear Feedback Shift Register, Linear Logic Function, Logic, Logic Function, LSB

M

M-Sequence, Machine Language, Magnetic Field, Man-in-the-Middle Attack, Mapping, Markov Process, Mathematical Cryptography, Maximal Length, MB, Mb, Mechanism, Mechanistic Cryptography, Mersenne Prime, Message Digest, Message Key, MITM, Mixing, Mixing Cipher, Mod 2, Mod 2 Polynomial, Mode, Modulo, Monadic, Monoalphabetic Substitution, Monographic, Multiple Encryption

N

Nominclator, Nominal, Nonlinearity, NOT, Null Hypothesis

O

Object Code, Objective, Octal, Octave, OFB, One Time Pad, One-To-One, One Way Diffusion, Onto, Opcode, Operating Mode, Opponent, OR, Order, Ordinal, Orthogonal, Orthogonal Latin Squares, OTP, Overall Diffusion

P

Padding, Password, Patent, Patent Infringement, Perfect Secrecy, Permutation, PGP, Physically Random, Pink Noise, Plaintext, Poisson Distribution, Polyalphabetic Combiner, Polyalphabetic Substitution, Polygram Substitution, Polygraphic, Polynomial, Polyphonic, Population, Population Estimation, Power, Primitive, Primitive Polynomial, Prime, Prior Art, PRNG, Process, Pseudorandom, Public Key Cipher

R

Random, Random Number Generator, Random Variable, Range, Really Random, Relay, Research Hypothesis, Resistor, Ring, Root, RMS, Root Mean Square, RNG, Round, RSA, Running Key

S

Salt, Sample, S-Box, Scalable, Secrecy, Secret Code, Secret Key Cipher, Security, Security Through Obscurity, Semiconductor, Semigroup, Session Key, Set, Shift Register, Shuffle, Sieve of Eratosthenes, Significance, Simple Substitution, Software, Source Code, State, Stationary Process, Statistic, Statistics, Steganography, Stochastic, Stream Cipher, Strength, Strict Avalanche Criterion (SAC), Subjective, Substitution, Substitution-Permutation, Substitution Table, Superencryption, Surjective, Switch, Switching Function, Symmetric Cipher, Symmetric Group, System, System Design

T

Table Selection Combiner, TEMPEST, Transformer, Transistor, Transposition, Trap Door, Triple DES, Truly Random, Trust, Truth Table, Type I Error, Type II Error

U

Unary, Unexpected Distance, Unicity Distance, Uniform Distribution

V

Variable Size Block Cipher, Voltage

W

Walsh Functions, Weight, Whitening White Noise Wire

X

XOR

Absolute

In the study of logic, something observed similarly by most observers, or something agreed upon, or which has the same value each time measured. Something not in dispute, unarguable, and independent of other state. As opposed to contextual.

AC

Alternating Current: Electrical power which repeatedly reverses direction of flow. As opposed to DC.

Generally used for power distribution because the changing current supports the use of transformers. Utilities can thus transport power at high voltage and low current, which minimize "ohmic" or I^2R losses. The high voltages are then reduced at power substations and again by pole transformers for delivery to the consumer.

Additive Combiner

An additive combiner uses numerical concepts similar to addition to mix multiple values into a single result.

One example is byte addition modulo 256, which simply adds two byte values, each in the range 0..255, and produces the remainder after division by 256, again a value in the byte range of 0..255. Subtraction is also an "additive" combiner.

Another example is bit-level exclusive-OR which is addition mod 2. A byte-level exclusive-OR is a polynomial addition.

Additive RNG

(Additive random number generator.) A LFSR-based RNG typically using multi-bit elements and integer addition (instead of XOR) combining. References include:

Knuth, D. 1981. *The Art of Computer Programming*, Vol. 2, *Seminumerical Algorithms*. 2nd ed. 26-31. Addison-Wesley: Reading, Massachusetts.

Marsaglia, G. and L. Tsay. 1985. Matrices and the Structure of Random Number Sequences. *Linear Algebra and its Applications*. 67: 147-156.

Advantages include:

- A long, mathematically proven cycle length.
- Especially efficient software implementations.
- Almost arbitrary initialization (some element must have its least significant bit set).
- A simple design which is easy to get right.

In addition, a vast multiplicity of independent cycles has the potential of confusing even a "quantum computer," should such a thing become possible.

```
For Degree-n Primitive, and Bit Width w

Total States:        2^nw
Non-Init States:     2^n(w-1)
Number of Cycles:    2^(n-1)(w-1)
Length Each Cycle:   (2^n-1)2^(w-1)
Period of LSB:       2^n-1
```

The binary addition of two bits with no carry input is just XOR, so the lsb of an Additive RNG has the usual maximal length period.

A degree-127 Additive RNG using 127 elements of 32 bits each has 2^{4064} unique states. Of these, 2^{3937} are disallowed by initialization (the lsb's are all "0") but this is just one unusable state out of 2^{127}. There are still 2^{3906} cycles which *each* have almost 2^{158} steps. (The Cloak2 stream cipher uses an Additive RNG with 9689 elements of 32 bits, and so has 2^{310048} unique states. These are mainly distributed among 2^{300328} different cycles with almost 2^{9720} steps each.)

Note that any LFSR, including the Additive RNG, is very weak when used alone. But when steps are taken to hide the sequence (such as using a jitterizer and Dynamic Substitution combining) the result can have significant strength.

Affine

Generally speaking, linear. Sometimes *affine* generalizes "linearity" to expressions of multiple independent variables, with only a single-variable expression being called "linear." From analytic and algebraic geometry.

Assume the flat plane defined by two arbitrary unit vectors $\mathbf{e}_1$, $\mathbf{e}_2$ and a common origin **O**; this is a coordinate "frame." Assume a grid of lines parallel to each frame vector, separated by unit lengths (a "metric" which may differ for each vector). If the vectors happen to be perpendicular, we have a Cartesian coordinate system, but in any case we can locate any point on the plane by its position on the grid.

An affine transformation can change the origin, the angle between the vectors, and unit vector lengths. Shapes in the original frame thus become "pinched," "squashed" or "stretched" images under the affine transformation. This same sort of thing generalizes to higher degree expressions.

The *Handbook of Mathematics* says that if $\mathbf{e}_1$, $\mathbf{e}_2$, $\mathbf{e}_3$ are linearly independent vectors, any vector **a** can be expressed uniquely in the form $\mathbf{a} = \mathbf{a}_1\mathbf{e}_1 + \mathbf{a}_2\mathbf{e}_2 + \mathbf{a}_3\mathbf{e}_3$ where the $\mathbf{a}_i$ are the *affine*

coordinates. (p.518)

The VNR Concise Encyclopedia of Mathematics says "All transformations that lead to a uniquely soluble system of linear equations are called *affine transformations*." (p.534)

Affine Boolean Function

A Boolean function which can be represented in the form:

$$a_n x_n + a_{n-1} x_{n-1} + \ldots + a_1 x_1 + a_0$$

where the operations are mod 2: addition is Exclusive-OR, and multiplication is AND.

Note that all of the variables x_i are to the first power only, and each coefficient a_i simply enables or disables its associated variable. The result is a single Boolean value, but the constant term a_0 can produce either possible output polarity.

Here are all possible 3-variable affine Boolean functions (each of which may be inverted by complementing the constant term):

```
 affine    truth table

      c    0  0  0  0  0  0  0  0
     x0    0  1  0  1  0  1  0  1
  x1       0  0  1  1  0  0  1  1
  x1+x0    0  1  1  0  0  1  1  0
x2         0  0  0  0  1  1  1  1
x2+   x0   0  1  0  1  1  0  1  0
  x2+x1    0  0  1  1  1  1  0  0
  x2+x1+x0 0  1  1  0  1  0  0  1
```

Alphabet

The set of symbols under discussion.

Alternative Hypothesis

In statistics, the statement formulated so that the logically contrary statement, the null hypothesis H_0 has a test statistic with a known distribution for the case when there is nothing unusual to detect. Also called the research hypothesis H_1, and logically identical to "NOT-H_0" or "H_0 is not true."

Amplifier

a component or device intended to sense a signal and produce a larger version of that signal. In general, any amplifying device is limited by available power, frequency response, and device maximums for voltage, current, and power dissipation.

Transistors are analog amplifiers which are basically linear over a reasonable range and so require DC power. In contrast, Relays are classically mechanical devices with direct metal-to-metal moving connections, and so can handle generally higher power and AC current.

Amplitude

The signal level, or height.

Analog

Pertaining to continuous values. As opposed to digital or discrete quantities.

AND

A Boolean logic function which is also mod 2 multiplication.

ASCII

A public code for converting between 7-bit values 0..127 (or 00..7f hex) and text characters. ASCII is an acronym for American Standard Code for Information Interchange.

DEC	HEX	CTRL	CMD	DEC	HEX	CHAR	DEC	HEX	CHAR	DEC	HEX	CHAR
0	00	^@	NUL	32	20	SPC	64	40	@	96	60	'
1	01	^A	SOH	33	21	!	65	41	A	97	61	a
2	02	^B	STX	34	22	"	66	42	B	98	62	b
3	03	^C	ETX	35	23	#	67	43	C	99	63	c
4	04	^D	EOT	36	24	$	68	44	D	100	64	d
5	05	^E	ENQ	37	25	%	69	45	E	101	65	e
6	06	^F	ACK	38	26	&	70	46	F	102	66	f
7	07	^G	BEL	39	27	'	71	47	G	103	67	g
8	08	^H	BS	40	28	(	72	48	H	104	68	h
9	09	^I	HT	41	29	)	73	49	I	105	69	i
10	0a	^J	LF	42	2a	*	74	4a	J	106	6a	j
11	0b	^K	VT	43	2b	+	75	4b	K	107	6b	k
12	0c	^L	FF	44	2c	,	76	4c	L	108	6c	l
13	0d	^M	CR	45	2d	-	77	4d	M	109	6d	m
14	0e	^N	SO	46	2e	.	78	4e	N	110	6e	n
15	0f	^O	SI	47	2f	/	79	4f	O	111	6f	o
16	10	^P	DLE	48	30	0	80	50	P	112	70	p
17	11	^Q	DC1	49	31	1	81	51	Q	113	71	q
18	12	^R	DC2	50	32	2	82	52	R	114	72	r
19	13	^S	DC3	51	33	3	83	53	S	115	73	s
20	14	^T	DC4	52	34	4	84	54	T	116	74	t
21	15	^U	NAK	53	35	5	85	55	U	117	75	u
22	16	^V	SYN	54	36	6	86	56	V	118	76	v
23	17	^W	ETB	55	37	7	87	57	W	119	77	w
24	18	^X	CAN	56	38	8	88	58	X	120	78	x
25	19	^Y	EM	57	39	9	89	59	Y	121	79	y
26	1a	^Z	SUB	58	3a	:	90	5a	Z	122	7a	z
27	1b	^[	ESC	59	3b	;	91	5b	[	123	7b	{
28	1c	^\	FS	60	3c	<	92	5c	\	124	7c	|
29	1d	^]	GS	61	3d	=	93	5d	]	125	7d	}
30	1e	^^	RS	62	3e	>	94	5e	^	126	7e	
31	1f	^_	US	63	3f	?	95	5f	_	127	7f	DEL

Associative

A dyadic operation in which two sequential operations on three arguments can first operate on either the first two or the last two arguments, producing the same result in either case: (a + b) + c = a + (b + c).

Also see: commutative and distributive.

Asymmetric Cipher

A public key cipher.

Attack

General ways in which a cryptanalyst may try to "break" or penetrate the secrecy of a cipher. These are **not** algorithms; they are just *approaches* as a starting place for constructing specific algorithms.

Classically, attacks were neither named nor classified; there was just: "here is a cipher, and here is the attack." And while this gradually developed into named attacks, there is no overall attack taxonomy. Currently, attacks are often classified by the information available to the attacker or *constraints* on the attack, and then by strategies which use the available information. Not only ciphers, but also cryptographic hash functions can be attacked, generally with very different strategies.

Informational Constraints

We are to attack a cipher which enciphers plaintext into ciphertext or deciphers the opposite way, under control of a key. The available information necessarily constrains our attack strategies.

- **Ciphertext Only:** We have only ciphertext to work with. Sometimes the statistics of the ciphertext provide insight and can lead to a break.
- **Known Plaintext:** We have some, or even an extremely large amount, of plaintext and the associated ciphertext.
- **Defined Plaintext:** We can submit arbitrary messages to be ciphered and capture the resulting ciphertext. (Also Chosen Plaintext and Adaptive Chosen Plaintext.)
- **Defined Ciphertext:** We can submit arbitrary messages to be deciphered and see the resulting plaintext. (Also Chosen Ciphertext and Adaptive Chosen Ciphertext.)
- **Chosen Key:** We can specify a change in any particular key bit, or some other relationship between keys.
- **Timing:** We can measure the duration of ciphering operations and use that to reveal the key or data.
- **Fault Analysis:** We can induce random faults into the ciphering machinery, and use those to expose the key.
- **Man-in-the-Middle:** We can subvert the routing capabilities of a computer network, and pose as the other side to each of the communicators. (Usually a key authentication attack on public key systems.)

Attack Strategies

The goal of an attack is to reveal some unknown plaintext, or the key (which will reveal the plaintext). An attack which succeeds with less effort than a brute-force search we call a break. An "academic" ("theoretical," "certificational") break may involve impractically large amounts of data or resources, yet still be called a "break" if the attack would be easier than brute force. (It is thus possible for a "broken" cipher to be much stronger than a cipher with a short key.) Sometimes the attack strategy is thought to be obvious, given a particular informational constraint, and is not further classified.

- **Brute Force** (also Exhaustive Key Search): Try to decipher ciphertext under every possible key until readable messages are produced. (Also "brute force" any searchable-size *part* of a cipher.)
- **Codebook** (the classic "codebreaking" approach): Collect a codebook of transformations between plaintext and ciphertext.
- **Differential Cryptanalysis:** Find a statistical correlation between key values and cipher transformations (typically the Exclusive-OR of text pairs), then use sufficient defined plaintext to develop the key.
- **Linear Cryptanalysis:** Find a linear approximation to the keyed S-boxes in a cipher, and use that to reveal the key.
- **Meet-in-the-Middle:** Given a two-level multiple encryption, search for the keys by collecting every possible result for enciphering a known plaintext under the first cipher, and deciphering the known ciphertext under the second cipher; then find the match.
- **Key Schedule:** Choose keys which produce known effects in different rounds.
- **Birthday** (usually a hash attack): Use the birthday paradox, the idea that it is much easier to find two values which match than it is to find a match to some particular value.
- **Formal Coding** (also Algebraic): From the cipher design, develop equations for the key in terms of known plaintext, then solve those equations.
- **Correlation**: In a stream cipher, distinguish between data and confusion, or between different confusion streams, from a statistical imbalance in a combiner.
- **Dictionary**: Form a list of the most-likely keys, then try those keys one-by-one (a way to improve brute force).
- **Replay**: Record and save some ciphertext blocks or messages (especially if the content is known), then re-send those blocks when useful.

Many attacks try to isolate unknown small components or aspects so they can be solved separately, a process known as divide and conquer. Also see: security.

Augmented Repetitions

When sampling with replacement, eventually we again find some object or value which has been found before. We call such an occurrence a "repetition." A value found exactly twice is a double, or "2-rep"; a value found three times is a triple or "3-rep," and so on.

For a known population, the number of repetitions expected at each level has long been understood to be a binomial expression. But if we are sampling in an attempt to *establish* the effective size of an unknown population, we have two problems:

1. The binomial equations which predict expected repetitions do not reverse well to predict population, and
2. Exact repetitions discard information and so are less accurate than we would like. For example, if we have a double and then find another of that value, we now have a triple, and one *less* double. So if we are using doubles to predict population, the occurrence of a triple influences the predicted population in exactly the wrong direction.

Fortunately, there is an unexpected and apparently previously unknown combinatoric relationship between the population and the number of combinations of occurrences of repeated values. This allows us to convert any number of triples and higher *n*-reps to the number of 2-reps which have the same probability. So if we have a double, and then get another of the same value, we have a triple, which we can convert into three 2-reps. The total number of 2-reps from all repetitions (the *augmented 2-reps* value) is then used to predict population.

We can relate the number of samples *s* to the population *N* through the expected number of augmented doubles *Ead*:

```
Ead(N,s) = s(s-1) / 2N .
```

This equation is **exact**, *provided* we interpret all the exact n-reps in terms of 2-reps. For example, a triple is interpreted as three doubles; the augmentation from 3-reps to 2-reps is (3 C 2) or 3. The augmented result is the sum of the contributions from all higher repetition levels:

```
       n    i
ad = SUM  ( ) r[i] .
     i=2   2
```

where *ad* is the number of augmented doubles, and *r[i]* is the exact repetition count at the *i*-th level.

And this leads to an equation for predicting population:

```
Nad(s,ad) = s(s-1) / 2 ad .
```

This predicts the population *Nad* as based on a mean value of augmented doubles *ad*. Clearly, we expect the number of samples to be far larger than the number of augmented doubles, but an error in the augmented doubles *ad* should produce a proportionally similar error in the predicted population *Nad*. We typically develop *ad* to high precision by averaging the results of many large trials.

However, since the trials should have approximately a simple Poisson distribution (which has only a single parameter), we could be a bit more clever and fit the results to the expected distribution, thus perhaps developing a bit more accuracy.

Also see the article: Estimating Population from Repetitions in Accumulated Random Samples, and the Population Estimation Worksheets in JavaScript page of the Ciphers By Ritter / JavaScript computation pages.

Authentication

One of the objectives of cryptography: Assurance that a message has not been modified in transit or storage (*message* authentication or message *integrity*). Also key authentication for public keys. Also user or source identification, which may verify the right to send the message in the first place.

Message Authentication

One form of message authentication computes a CRC hash across the plaintext data, and appends the CRC remainder (or *result*) to the plaintext data: this adds a computed redundancy to an arbitrary message. The CRC result is then enciphered along with the data. When the message is deciphered, if a second CRC operation produces the same result, the message can be assumed unchanged.

Note that a CRC is a fast, linear hash. Messages with particular CRC result values can be constructed rather easily. However, if the CRC is hidden behind strong ciphering, an Opponent is unlikely to be able to change the CRC value systematically or effectively. In particular, this means that the CRC value will need more protection than a simple exclusive-OR stream cipher or the exclusive-OR approach to handling short last blocks in a block cipher.

A similar approach to message authentication uses a nonlinear cryptographic hash function. These also add a computed redundancy to the message, but generally require significantly more computation than a CRC. It is thought to be exceedingly difficult to construct messages with a particular cryptographic hash result, so the hash result perhaps need not be hidden by encryption.

One form of cryptographic hash is DES CBC mode: using a key different than that used for encryption, the final block of ciphertext is the hash of the message. This obviously doubles the computation when both encryption and authentication are needed. And since any cryptographic hash is vulnerable to birthday attacks, the small 64-bit block size implies that we should be able to find two different messages with the same hash value by constructing and hashing "only" about 2^{32} different messages.

Another approach to message authentication is to use an authenticating block cipher; this is often a block cipher which has a large block, with some "extra data" inserted in an "authentication field" as part of the plaintext before enciphering each block. The "extra data" can be some transformation of the key, the plaintext, and/or a sequence number. This essentially creates a homophonic block cipher: If we know the key, many different ciphertexts will produce the same plaintext field, but only one of those will have the correct authentication field.

The usual approach to authentication in a public key cipher is to encipher with the private key. The resulting ciphertext can then be deciphered by the public key, which anyone can know. Since even the wrong key will produce a "deciphered" result, it is also necessary to identify the resulting plaintext as a valid message; in general this will also require redundancy in the form of a hash value in the plaintext. The process provides no secrecy, but only a person with access to the private key could have enciphered the message.

User Authentication

The classical approach to user authentication is a password; this is "something you know." One can also make use of "something you have" (such as a secure ID card), or "something you are" (biometrics).

The classic problem with passwords is that they must be remembered by ordinary people, and so carry a limited amount of uniqueness. Easy-to-remember passwords are often common language phrases, and so often fall to a dictionary attack. More modern approaches involve using a Diffie-Hellman key exchange, *plus* the password, thus minimizing exposure to a dictionary attack. This does require a program on the user end, however.

Key Authentication

In secret key ciphers, key authentication is *inherent* in secure key distribution.

In public key ciphers, public keys are exposed and often delivered insecurely. But someone who uses the wrong key may unknowingly have "secure" communications with an Opponent, as in a man-in-the-middle attack. It is thus absolutely crucial that public keys be authenticated or *certified* as a separate process. Normally this implies the need for a Certification Authority or CA.

Authenticating Block Cipher

A block cipher mechanism which inherently contains an authentication value or field.

Autokey

A cipher whose key is produced by message data. One common form is "ciphertext feedback," where ciphertext is "fed back" into the state of the random number generator used to produce the confusion sequence for a stream cipher.

Avalanche

The observed property of a block cipher constructed in layers or "rounds" with respect to a tiny change in the input. The change of a single input bit generally produces multiple bit-changes after one round, many more bit-changes after another round, until, eventually, about half of the block will change. An analogy is drawn to an avalanche in snow, where a small initial effect can lead to a dramatic result. As originally described by Feistel:

> "As the input moves through successive layers the pattern of 1's generated is amplified and results in an unpredictable avalanche. In the end the final output will have, on average, half 0's and half 1's . . . ." [p.22]

Feistel, H. 1973. Cryptography and Computer Privacy. *Scientific American.* 228(5): 15-23.

Also see mixing, diffusion, overall diffusion, strict avalanche criterion, complete, S-box, and the bit changes section of the Ciphers By Ritter / JavaScript computation pages.

Avalanche Effect

The result of avalanche. As described by Webster and Tavares:

> "For a given transformation to exhibit the avalanche effect, an average of one half of the output bits should change whenever a single input bit is complemented." [p.523]

Webster, A. and S. Tavares. 1985. On the Design of S-Boxes. *Advances in Cryptology -- CRYPTO '85.* 523-534.

Also see the bit changes section of the Ciphers By Ritter / JavaScript computation pages.

Back Door

A cipher design fault, planned or accidental, which allows the apparent strength of the design to be easily avoided by those who know the trick. When the design background of a cipher is kept secret, a back door is often suspected. Similar to trap door.

Balance

A term used in S-box and Boolean function analysis. As described by Lloyd:

> "A function is balanced if, when all input vectors are equally likely, then all output vectors are equally likely."

Lloyd, S. 1990. Properties of binary functions. *Advances in Cryptology -- EUROCRYPT '90.* 124-139.

There is some desire to generalize this definition to describe multiple-input functions. (Is a function "balanced" if, for one value on the first input, all output values can be produced, but for another value on the first input, only *some* output values are possible?) Presumably a two-input balanced function would be balanced for either input fixed at any value, which would essentially be a Latin square or a Latin square combiner.

Balanced Block Mixer

A process or any implementation (for example, hardware, computer software, hybrids, or the like) for performing Balanced Block Mixing.

Balanced Block Mixing

The block mixing mechanism described in U.S. Patent 5,623,549 (see the BBM articles on the Ciphers By Ritter page).

A Balanced Block Mixer is an *m*-input-port *m*-output-port mechanism with various properties:

1. The overall mapping is one-to-one and invertible: Every possible input value (over all ports) to the mixer produces a different output value (including all ports), and every possible output value is produced by a different input value;
2. Each output port is a function of every input port;
3. Any change to any one of the input ports will produce a change to every output port;
4. Stepping any one input port through all possible values (while keeping the other input ports fixed) will step every output port through all possible values.

If we have a two port mixer, with input ports labeled *A* and *B*, output ports labeled *X* and *Y*, and some irreducible mod 2 polynomial *p* of degree appropriate to the port size, a Balanced Block Mixer is formed by the equations:

$$X = 3A + 2B \pmod 2 \pmod p,$$
$$Y = 2A + 3B \pmod 2 \pmod p.$$

This particular BBM is a self-inverse or involution, and so can be used without change whether enciphering or deciphering. One possible value for *p* for mixing 8-bit values is 100011011.

Balanced Block Mixing functions probably should be thought of as orthogonal Latin squares. For example, here is a tiny nonlinear "2-bit" BBM:

```
3 1 2 0   0 3 2 1       30  13  22  01
```

```
0 2 1 3   2 1 0 3   =   02  21  10  33
1 3 0 2   1 2 3 0       11  32  03  20
2 0 3 1   3 0 1 2       23  00  31  12
```

Suppose we wish to mix (1,3); 1 selects the second row up in both squares, and 3 selects the rightmost column, thus selecting (2,0) as the output. Since there is only one occurrence of (2,0) among all entry pairs, this discrete mixing function is reversible, as well as being balanced on both inputs.

Cryptographic advantages of balanced block mixing include the fact that each output is always balanced with respect to either input, and that no information is lost in the mixing. This allows us to use balanced block mixing as the "butterfly" operations in a fast Walsh-Hadamard transform or the well-known FFT. By using the mixing patterns of these transforms, we can mix 2^n elements such that each input is guaranteed to affect each and every output in a balanced way. And if we use keying to generate the tables, we can have a way to mix huge blocks in small nonlinear mixing tables with overall mixing guarantees.

Also see Mixing Cipher, Dynamic Substitution Combiner, Variable Size Block Cipher, and the Active Balanced Block Mixing in JavaScript page of the Ciphers By Ritter / JavaScript computation pages.

Balanced Combiner

In the context of cryptography, a combiner mixes two input values into a result value. A balanced combiner must provide a balanced relationship between each input and the result.

In a *statically-balanced* combiner, any particular result value can be produced by any value on one input, simply by selecting some appropriate value for the other input. In this way, knowledge of only the output value provides no information -- not even statistical information -- about either input.

The common examples of cryptographic combiner, including byte exclusive-OR (mod 2 polynomial addition), byte addition (integer addition mod 256), or other "additive" combining, are perfectly balanced. Unfortunately, these simple combiners are also very weak, being inherently linear and without internal state.

A Latin square combiner is an example of a statically-balanced reversible nonlinear combiner with massive internal state. A Dynamic Substitution Combiner is an example of a dynamically or statistically-balanced reversible nonlinear combiner with substantial internal state.

Base-64

A public code for converting between 6-bit values 0..63 (or 00..3f hex) and text symbols accepted by most computers:

```
      0  1  2  3  4  5  6  7  8  9  a  b  c  d  e  f

0     A  B  C  D  E  F  G  H  I  J  K  L  M  N  O  P
1     Q  R  S  T  U  V  W  X  Y  Z  a  b  c  d  e  f
2     g  h  i  j  k  l  m  n  o  p  q  r  s  t  u  v
3     w  x  y  z  0  1  2  3  4  5  6  7  8  9  +  /

use "=" for padding
```

Bel

The base-10 logarithm of the ratio of two power values (which is also the same as the difference between the log of each power value). The basis for the more-common term decibel: One bel equals 10 decibels.

Bent Function

A bent function is a Boolean function whose fast Walsh transform has the same absolute value in each term (except, possibly, the zeroth). This means that the bent function has the same distance from every possible affine Boolean function.

We can do FWT's in "the bottom panel" at the end of Active Boolean Function Nonlinearity Measurement in JavaScript page of the Ciphers By Ritter / JavaScript computation pages.

Here is every bent sequence of length 4, first in {0,1} notation, then in {1,-1} notation, with their FWT results:

```
bent {0,1}       FWT            bent {1,-1}       FWT

0  0  0  1    1 -1 -1  1         1  1  1 -1    2  2  2 -2
0  0  1  0    1  1 -1 -1         1  1 -1  1    2 -2  2  2
0  1  0  0    1 -1  1 -1         1 -1  1  1    2  2 -2  2
1  0  0  0    1  1  1  1        -1  1  1  1    2 -2 -2 -2
1  1  1  0    3  1  1 -1        -1 -1 -1  1   -2 -2 -2  2
1  1  0  1    3 -1  1  1        -1 -1  1 -1   -2  2 -2  2
1  0  1  1    3  1 -1  1        -1  1 -1 -1   -2 -2  2 -2
0  1  1  1    3 -1 -1 -1         1 -1 -1 -1   -2  2  2  2
```

These sequences, like all true bent sequences, are **not** balanced, and the zeroth element of the {0,1} FWT is the number of 1's in the sequence.

Here are some bent sequences of length 16:

```
bent {0,1}    0 1 0 0   0 1 0 0   1 1 0 1   0 0 1 0
 FWT          6,-2,2,-2,2,-2,2,2,-2,-2,2,-2,-2,2,-2,-2
bent {1,-1}   1 -1 1 1   1 -1 1 1   -1 -1 1 -1   1 1 -1 1
 FWT          4,4,-4,4,-4,4,-4,-4,4,4,-4,4,4,-4,4,4

bent {0,1}    0 0 1 0   0 1 0 0   1 0 0 0   1 1 1 0
 FWT          6,2,2,-2,-2,2,-2,2,-2,-2,-2,-2,2,2,-2,-2
bent {1,-1}   1 1 -1 1   1 -1 1 1   -1 1 1 1   -1 -1 -1 1
 FWT          4,-4,-4,4,4,-4,4,-4,4,4,4,4,-4,-4,4,4
```

Bent sequences are said to have the highest possible uniform nonlinearity. But, to put this in perspective, recall that we *expect* a random sequence of 16 bits to have 8 bits different from any particular sequence, linear or otherwise. That is also the *maximum possible* nonlinearity, and here we actually *get* a nonlinearity of 6.

There are various more or less complex constructions for these sequences. In most cryptographic uses, bent sequences are modified slightly to achieve balance.

Bernoulli Trials

In statistics, observations or sampling with replacement which has exactly two possible outcomes, typically called "success" and "failure." Bernoulli trials have these characteristics:

- Each trial is independent,
- Each outcome is determined only by chance, and
- The probability of success is fixed.

Bernoulli trials have a Binomial distribution.

Bijective

A mapping f: *X* -> *Y* which is both one-to-one and onto. For each unique *x* in *X* there is corresponding unique *y* in *Y*. An invertible mapping function.

Binary

From the Latin for "dual" or "pair." Dominantly used to indicate "base 2": The numerical representation in which each digit has an alphabet of only two symbols: 0 and 1. This is just one particular coding or representation of a value which might otherwise be represented (with the exact same value) as octal (base 8), decimal (base 10), or hexadecimal (base 16). Also see bit and Boolean.

Possibly also the confusing counterpart to unary when describing the number of inputs or arguments to a function, but dyadic is almost certainly a better choice.

Binomial Distribution

In statistics, the probability of finding exactly *k* successes in *n* independent Bernoulli trials, when each trial has success probability *p*:

$$P(k,n,p) = \binom{n}{k} p^{k} (1-p)^{n-k}$$

This ideal distribution is produced by evaluating the probability function for all possible *k*, from 0 to *n*.

If we have an experiment which we think *should* produce a binomial distribution, and then repeatedly and systematically find very improbable test values, we may choose to reject the null hypothesis that the experimental distribution is in fact binomial.

Also see the binomial section of the Ciphers By Ritter / JavaScript computation pages.

Birthday Attack

A form of attack in which it is necessary to obtain two identical values from a large population. The "birthday" part is the realization that it is far easier to find an arbitrary matching pair than to match any particular value. Often a hash attack.

Also see: birthday paradox.

Birthday Paradox

The apparent paradox that, in a schoolroom of only 23 students, there is a 50 percent probability that at least two will have the same birthday. The "paradox" is that we have an even chance of success with at most 23 different days represented.

The "paradox" is resolved by noting that we have a 1/365 chance of success for each possible *pairing* of students, and there are 253 possible pairs or combinations of 23 things taken 2 at a time. (To count the number of pairs, we can choose any of the 23 students as part of the pair, then any of the 22 remaining students as the other part. But this counts each pair twice, so we have 23 * 22 / 2 = 253 different pairs.)

We can compute the overall probability of success from the probability of *failure* (1 - 1/365 = 0.99726) multiplied by itself for each pair. The overall probability of failure is thus 0.99726^{253} (0.99726 to the 253rd power) or 0.4995. So the success probability for 253 pairs is 0.5005.

We can relate the probability of finding at least one "double" of some birthday (Pd) to the expected number of doubles (Ed) as:

$$Pd = 1 - e^{-Ed} ,$$

so

$$Ed = -Ln(1 - Pd)$$

and

$$365 * -Ln(0.5) = 365 * 0.693 = 253 .$$

Also see: Estimating Population from Repetitions in Accumulated Random Samples, my "birthday" article.

Bit

A contraction of "binary digit." The smallest possible unit of information. A Boolean value: True or False; Yes or No; one or zero; Set or Cleared. Virtually all information to be communicated or stored digitally is coded in some way which fundamentally relies on individual bits. Alphabetic characters are often stored in eight bits, which is a byte.

Block

Some amount of data treated as a single unit. For example, the DES block cipher has a 64-bit block. So DES ciphers 64 bits (8 bytes or typically 8 ASCII characters) at once.

A 64-bit block supports 2^{64} or about 1.8 x 10^{19} block values or code values. Each different permutation of those values can be considered a complete code. A block cipher has the ability to select from among many such codes using a key.

It is not normally possible to block-cipher just a single bit or a single byte of a block. An arbitrary stream of data can always be partitioned into one or more fixed-size blocks, but it is likely that at least one block will not be completely filled. Using fixed-size blocks generally means that the associated system must support data expansion in enciphering, if only by one block. Handling even minimal data expansion may be difficult in some systems.

Block Cipher

A cipher which requires the accumulation of data (in a block) before ciphering can complete. Other than simple transposition ciphers, this seems to be the province of ciphers designed to *emulate* a keyed simple substitution with a table of size far too large to realize. A block cipher operates on a block of data (for example, multiple bytes) in a single ciphering, as opposed to a stream cipher, which operates on bytes or bits as they occur. Block ciphers can be called "codebook-style" ciphers. Also see Variable Size Block Cipher.

A block cipher is a transformation between plaintext block values and ciphertext block values, and is thus an emulated simple substitution on huge block-wide values. Within a particular block size, both plaintext and ciphertext have the same set of possible values, and when the ciphertext values have the same ordering as the plaintext, ciphering is obviously ineffective. So *effective* ciphering depends upon *re-arranging* the ciphertext values from the plaintext ordering, and this is a permutation of the plaintext values. A block cipher is keyed by constructing a *particular* permutation of ciphertext values.

Block Cipher Data Diffusion

In an ideal block cipher, changing even a single bit of the input block will change all bits of the ciphertext result, each with independent probability 0.5. This means that about half of the bits in the output will change for any different input block, even for differences of just one bit. This is overall diffusion and is present in a block cipher, but not in a stream cipher. Data diffusion is a simple consequence of the keyed invertible simple substitution nature of the ideal block cipher.

Improper diffusion of data throughout a block cipher can have serious strength implications. One of the functions of data diffusion is to hide the different effects of different internal components. If these effects are not in fact hidden, it may be possible to attack each component separately, and break the whole cipher fairly easily.

Partitioning Messages into Fixed Size Blocks

A large message can be ciphered by partitioning the plaintext into blocks of a size which can be ciphered. This essentially creates a stream meta-cipher which repeatedly uses the same block cipher transformation. Of course, it is also possible to re-key the block cipher for each and every block ciphered, but this is usually expensive in terms of computation and normally unnecessary.

A message of arbitrary size can always be partitioned into some number of whole blocks, with possibly some space remaining in the final block. Since partial blocks cannot be ciphered, some random padding can be introduced to fill out the last block, and this naturally expands the ciphertext. In this case it may also be necessary to introduce some sort of structure which will indicate the number of valid bytes in the last block.

Block Partitioning without Expansion

Proposals for using a block cipher supposedly *without* data expansion may involve creating a tiny stream cipher for the last block. One scheme is to re-encipher the ciphertext of the preceding block, and use the result as the confusion sequence. Of course, the cipher designer still needs to address the situation of files which are so short that they *have* no preceding block. Because the one-block version is *in fact* a stream cipher, we must be very careful to never re-use a confusion sequence. But when we only *have* one block, there *is* no prior block to change as a result of the data. In this case, ciphering several very short files could expose those files quickly. Furthermore, it is dangerous to encipher a CRC value in such a block, because exclusive-OR enciphering is transparent to the field of mod 2 polynomials in which the CRC operates. Doing this could allow an Opponent to adjust the message CRC in a known way, thus avoiding authentication exposure.

Another proposal for eliminating data expansion consists of ciphering blocks until the last short block, then re-positioning the ciphering window to end at the last of the data, thus re-ciphering part of the prior block. This is a form of chaining and establishes a sequentiality requirement which requires that the last block be deciphered *before* the next-to-the-last block. Or we can make enciphering inconvenient and deciphering easy, but one way will be a problem. And this approach cannot handle very short messages: its minimum size is one block. Yet any general-purpose ciphering routine *will* encounter short messages. Even worse, if we have a short message, we still need to somehow indicate the correct length of the message, and this must expand the message,

as we saw before. Thus, overall, this seems a somewhat dubious technique.

On the other hand, it does show a way to chain blocks for authentication in a large-block cipher: We start out by enciphering the data in the first block. Then we position the next ciphering to start *inside* the ciphertext of the previous block. Of course this would mean that we would have to decipher the message in reverse order, but it would also propagate any ciphertext changes through the end of the message. So if we add an authentication field at the end of the message (a keyed value known on both ends), and that value is recovered upon deciphering (this will be the first block deciphered) we can authenticate the whole message. But we still need to handle the last block padding problem and possibly also the short message problem.

Block Size and Plaintext Randomization

Ciphering raw plaintext data can be dangerous when the cipher has a small block size. Language plaintext has a strong, biased distribution of symbols and ciphering raw plaintext would effectively reduce the number of possible plaintexts blocks. Worse, some plaintexts would be vastly more probable than others, and if some known plaintext were available, the most-frequent blocks might already be known. In this way, small blocks can be vulnerable to classic codebook attacks which build up the ciphertext equivalents for many of the plaintext phrases. This sort of attack confronts a particular block size, and for these attacks Triple-DES is no stronger than simple DES, because they both have the same block size.

The usual way of avoiding these problems is to randomize the plaintext block with an operating mode such as CBC. This can ensure that the plaintext data which is actually ciphered is evenly distributed across all possible block values. However, this also requires an IV which thus expands the ciphertext.

Another approach is to apply data compression to the plaintext before enciphering. If this is to be used *instead* of plaintext randomization, the designer must be very careful that the data compression does not contain regular features which could be exploited by The Opponents.

An alternate approach is to use blocks of sufficient size for them to be expected to have a substantial amount of uniqueness or "entropy." If we expect plaintext to have about one bit of entropy per byte of text, we might want a block size of at least 64 bytes before we stop worrying about an uneven distribution of plaintext blocks. This is now a practical block size.

Boolean

TRUE or FALSE; one bit of information.

Boolean Function

A function which produces a Boolean result. The individual output bits of an S-box can each be considered to be separate Boolean functions.

Boolean Function Nonlinearity

The number of bits which must change in the truth table of a Boolean function to reach the closest affine Boolean function. This is the Hamming distance from the closest "linear" function.

Typically computed by using a fast Walsh-Hadamard transform on the Boolean-valued truth table of the function. This produces the unexpected distance to every possible affine Boolean function (of the given length). Scanning those results for the maximum value implies the minimum distance to some particular affine sequence.

Especially useful in S-box analysis, where the nonlinearity for the table is often taken to be the minimum of the nonlinearity values computed for each output bit.

Also see the Active Boolean Function Nonlinearity Measurement in JavaScript page of the Ciphers By Ritter / JavaScript computation pages.

Boolean Logic

The logic which applies to variables which have only two possible values. Also the digital hardware devices which realize such logic, and are used to implement a electronic digital computers.

Boolean Mapping

A mapping of some number *n* Boolean variables into some number *m* Boolean results. For example, an S-box.

Break

The result of a successful cryptanalytic attack. To destroy the advantage of a cipher in hiding information.

A cipher is "broken" when the information in a message can be extracted without the key, or when the key itself can be recovered. The strength of a cipher can be considered to be the minimum effort required for a break, by any possible attack. A break is particularly significant when the work involved need not be repeated on every message.

The use of the term "break" can be misleading when an impractical amount of work is required to achieve the break. This case might be better described a "theoretical" or "certificational" *weakness*.

Block Size

The amount of data in a block. For example, the size of the DES block is 64 bits or 8 bytes or 8 octets.

Brute Force Attack

A form of attack in which each possibility is tried until success is obtained. Typically, a ciphertext is deciphered under different keys until plaintext is recognized. On average, this may take about half as many decipherings as there are keys.

Recognizing plaintext may or may not be easy. Even when the key length of a cipher is sufficient to prevent brute force attack, that key will be far too small to produce every possible plaintext from a given ciphertext (see perfect secrecy). Combined with the fact that language is redundant, this means that very few of the decipherings will be words in proper form. Of course, if the plaintext is not language, but is instead computer code, compressed text, or even ciphertext from another cipher, recognizing a correct deciphering can be difficult.

Brute force is the obvious way to attack a cipher, and the way any cipher can be attacked, so ciphers are designed to have a large enough keyspace to make this much too expensive to use in practice. Normally, the design strength of a cipher is based on the cost of a brute-force attack.

Bug

Technical slang for "error in design or implementation." An unexpected system flaw. Debugging is a normal part of system development and interactive system design.

Byte

A collection of eight bits. Also called an "octet." A byte can represent 256 different values or symbols. The common 7-bit ASCII codes used to represent characters in computer use are generally stored in a byte; that is, one byte per character.

Capacitor

A basic electronic component which acts as a reservoir for electrical power in the form of voltage. A capacitor thus acts to "even out" the voltage across its terminals, and to "conduct" voltage changes from one terminal to the other. A capacitor "blocks" DC and conducts AC in proportion to frequency. Capacitance is measured in Farads: A current of 1 Amp into a capacitance of 1 Farad produces a voltage change of 1 Volt per Second across the capacitor.

Typically, two conductive "plates" or metal foils separated by a thin insulator, such as air, paper, or ceramic. An electron charge on one plate attracts the opposite charge on the other plate, thus "storing" charge. A capacitor can be used to collect a small current over long time, and then release a high current for a short time, as used in a camera strobe or "flash."

Also see inductor and resistor.

CBC

CBC or Cipher Block Chaining is an operating mode for block ciphers. CBC mode is essentially a crude meta-stream cipher which streams block transformations.

In CBC mode the ciphertext value of the preceding block is exclusive-OR combined with the plaintext value for the current block. This has the effect of distributing the combined block values evenly among all possible block values, and so prevents codebook attacks.

On the other hand, ciphering the *first* block generally requires an IV or initial value to start the process. The IV necessarily expands the ciphertext, which may or may not be a problem. And the IV must be dynamically random-like so that statistics cannot be developed on the first block of each message sent under the same key.

In CBC mode, each random-like confusing value is the ciphertext from each previous block. Clearly this ciphertext is exposed to The Opponent, so there would seem to be little benefit associated with hiding the IV, which is just the first of these values. But if The Opponent knows the first sent plaintext, and can intercept and change the message IV, The Opponent can manipulate the first block of received plaintext. Because the IV does not represent a message enciphering, manipulating this value does not also change any previous block.

Accordingly, the IV may be sent enciphered or may be specifically authenticated in some way. Alternately, the complete body of the plaintext message may be authenticated, often by a CRC. The CRC remainder should be block ciphered, perhaps as part of the plaintext.

c.d.f.

In statistics, *cumulative distribution function.* A function which gives the probability of obtaining a particular value or lower.

CFB

CFB or Ciphertext FeedBack is an operating mode for a block cipher.

CFB is closely related to OFB, and is intended to provide some of the characteristics of a stream cipher from a block cipher. CFB generally forms an autokey stream cipher. CFB is a way of using a block cipher to form a random number generator. The resulting pseudorandom confusion sequence can be combined with data as in the usual stream cipher.

CFB assumes a shift register of the block cipher block size. An IV or initial value first fills the register, and then is ciphered. Part of the result, often just a single byte, is used to cipher data, and the resulting ciphertext is also shifted into the register. The new register value is ciphered, producing another confusion value for use in stream ciphering.

One disadvantage of this, of course, is the need for a full block-wide ciphering operation, typically for each data byte ciphered. The advantage is the ability to cipher individual characters, instead of requiring accumulation into a block before processing.

Chain

An operation repeated in a sequence, such that each result depends upon the previous result, or an initial value. One example is the CBC operating mode.

Chaos

The unexpected ability to find numerical relationships in physical processes formerly considered random. Typically these take the form of iterative applications of fairly simple computations. In a chaotic system, even tiny changes in state eventually lead to major changes in state; this is called "sensitive dependence on initial conditions." It has been argued that every good computational random number generator is "chaotic" in this sense.

In physics, the "state" of an analog physical system cannot be fully measured, which always leaves some remaining uncertainty to be magnified on subsequent steps. And, in many cases, a physical system may be slightly affected by thermal noise and thus continue to accumulate new information into its "state."

In a computer, the state of the digital system is explicit and complete, and there is no uncertainty. No noise is accumulated. All operations are completely deterministic. This means that, in a computer, even a "chaotic" computation is completely predictable and repeatable.

Chi-Square

In statistics, a goodness of fit test used for comparing two distributions. Mainly used on nominal and ordinal measurements. Also see: Kolmogorov-Smirnov.

In the usual case, many independent samples are counted by category or separated into value-range "bins." The reference distribution gives us the the number of values to expect in each bin. Then we compute a X^2 test statistic related to the difference between the distributions:

```
X² = SUM( SQR(Observed[i] - Expected[i]) / Expected[i] )
```

("SQR" is the squaring function, and we require that each expectation not be zero.) Then we use a tabulation of chi-square statistic values to look up the probability that a particular X^2 value or lower (in the c.d.f.) would occur by random sampling if both distributions were the same. The statistic also depends upon the "degrees of freedom," which is almost always one less than the final number of bins. See the chi-square section of the Ciphers By Ritter / JavaScript computation pages.

The c.d.f. percentage for a particular chi-square value is the area of the statistic distribution to the left of the statistic value; this is the probability of obtaining that statistic value *or less* by random selection when testing two distributions which are exactly the same. Repeated trials which randomly sample two identical distributions should produce about the same number of X^2 values in each quarter of the distribution (0% to 25%, 25% to 50%, 50% to 75%, and 75% to 100%). So if we repeatedly find only very high percentage values, we can assume that we are probing different distributions. And even a single very high percentage value would be a matter of some interest.

Any statistic probability can be expressed either as the proportion of the area to the *left* of the statistic value (this is the "cumulative distribution function" or c.d.f.), or as the area to the *right* of the value (this is the "upper tail"). Using the upper tail representation for the X^2 distribution can make sense because the usual chi-squared test is a "one tail" test where the decision is always made on the upper tail. But the "upper tail" has an opposite "sense" to the c.d.f., where higher statistic values always produce higher percentage values. Personally, I find it helpful to describe all statistics by their c.d.f., thus avoiding the use of a wrong "polarity" when interpreting any particular statistic. While it is easy enough to convert from the c.d.f. to the complement or vise versa (just subtract from 1.0), we can base our arguments on either form, since the statistical implications are the same.

It is often unnecessary to use a statistical test if we just want to know whether a function is producing something like the expected distribution: We can *look* at the binned values and generally get a good idea about whether the distributions change in similar ways at similar places. A good rule-of-thumb is to expect chi-square totals similar to the number of bins, but distinctly different distributions often produce huge totals far beyond the values in any table, and computing an exact probability for such cases is simply irrelevant. On the other hand, it can be very useful to perform 20 to 40 independent experiments to look for a reasonable statistic distribution, rather than simply making a "yes / no" decision on the basis of what might turn out to be a rather unusual result.

Since we are accumulating *discrete* bin-counts, any fractional expectation will always differ from any actual count. For example, suppose we expect an even distribution, but have many bins and so only accumulate enough samples to observe about 1 count for every 2 bins. In this situation, the absolute best sample we could hope to see would be something like (0,1,0,1,0,1,...), which would represent an even, balanced distribution over the range. But even in this best possible case we would still be off by half a count in each and every bin, so the chi-square result would not properly characterize this best possible sequence. Accordingly, we need to accumulate enough samples so that the quantization which occurs in binning does not appreciably affect the accuracy of the result. Normally I try to expect at least 10 counts in each bin.

But when we have a reference distribution that trails off toward zero, *inevitably* there will be some bins with few counts. Taking more samples will just expand the range of bins, some of which will be lightly filled in any case. We can avoid quantization error by summing both the observations and expectations from multiple bins, until we get a reasonable expectation value (again, I like to see 10 counts or more). In this way, the "tails" of the distribution can be more properly (and legitimately) characterized.

Cipher

In general, a key-selected secret transformation between plaintext and ciphertext. Specifically, a secrecy mechanism or process which operates on individual characters or bits independent of semantic content. As opposed to a secret code, which generally operates on words, phrases or sentences, each of which may carry some amount of complete meaning. Also see: cryptography, block cipher, stream cipher, a cipher taxonomy, and substitution.

A good cipher can transform secret information into a multitude of different intermediate forms, each of which represents the original information. *Any* of these intermediate forms or ciphertexts can be produced by ciphering the information under a particular key value. The intent is that the original information only be exposed by *one* of the many possible keyed interpretations of that ciphertext. Yet the correct interpretation is available merely by deciphering under the appropriate key.

A cipher appears to reduce the protection of secret information to enciphering under some key, and then keeping that key secret. This is a great reduction of effort and potential exposure, and is much like keeping your valuables in your house, and then locking the door when you leave. But there are also similar limitations and potential problems.

With a good cipher, the resulting ciphertext can be stored or transmitted otherwise exposed without also exposing the secret information hidden inside. This means that ciphertext can be stored in, or transmitted through, systems which have no secrecy protection. For transmitted information, this also means that the cipher itself must be distributed in multiple places, so in general the cipher cannot be assumed to be secret. With a good cipher, only the deciphering key need be kept secret.

A Cipher Taxonomy

For the analysis of cipher operation it is useful to collect ciphers into groups based on their functioning (or *intended* functioning). The goal is to group ciphers which are *essentially similar*, so that as we gain an understanding of one cipher, we can apply that understanding to others in the same group. We thus classify *not* by the components which make up the cipher, but instead on the "black-box" operation of the cipher itself.

We seek to hide distinctions of size, because *operation* is independent of size, and because size effects are usually straightforward. We thus classify serious block ciphers as keyed simple substitution, just like newspaper amusement ciphers, despite their obvious differences in strength and construction. This allows us to compare the results from an ideal tiny cipher to those from a large cipher construction; the grouping thus can provide *benchmark* characteristics for measuring large cipher constructions.

We *could* of course treat each cipher as an entity unto itself, or relate ciphers by their dates of discovery, the tree of developments which produced them, or by known strength. But each of these criteria is more or less limited to telling us "this cipher is what it is." We already know that. What we *want* to know is what other ciphers function in a similar way, and then whatever is known about *those* ciphers. In this way, every cipher need not be an island unto itself, but instead can be judged and compared in a related community of similar techniques.

Our primary distinction is between ciphers which handle all the data at once (block ciphers), and those which handle some, then some more, then some more (stream ciphers). We thus see the usual repeated use of a block cipher as a stream *meta-cipher* which has the block cipher as a component. It is also possible for a stream cipher to be re-keyed or re-originate frequently, and so appear to operate on "blocks." Such a cipher, however, would not have the overall diffusion we normally associate with a block cipher, and so might usefully be regarded as a stream meta-cipher with a stream cipher component.

The goal is not to give each cipher a label, but instead to seek insight. Each cipher in a particular general class carries with it the consequences of that class. And because these groupings ignore

size, we are free to generalize from the small to the large and so predict effects which may be unnoticed in full-size ciphers.

. **BLOCK CIPHER**

A block cipher *requires* the accumulation of some amount of data or multiple data elements for ciphering to complete. (Sometimes stream ciphers accumulate data for convenience, as in cylinder ciphers, which nevertheless logically cipher each character independently.)

(Note that this definition is somewhat broader than the now common understanding of a huge, and thus *emulated*, Simple Substitution. But there are ciphers which require blocked plaintext and which do *not* emulate Simple Substitution, and calling these something other than "block" ciphers negates the advantage of a taxonomy.)

1. **SUBSTITUTION CIPHER**
   - A "codebook" or "simple substitution."
   - Each code value becomes a distinguishable element. Thus, substitution generally converts a collection of independent elements to a single related unit.
   - Keying constitutes a permutation or re-arrangement of the fixed set of possible code values.
   - Avalanche or data diffusion is a natural consequence of an arbitrary selection among all possible code values.
   - The usual complete binary substitution distributes bit-changes between code values binomially, and this effect can be sampled and examined statistically.
   - Avalanche is two-way diffusion in the sense that "later" plaintext can change "earlier" ciphertext.
   - A conventional block cipher is built from small components with a design intended to *simulate* a substitution table of a size vastly larger than anything which could be practically realized.

   . **Transposition Cipher**
     - Clearly, it is necessary for all message elements which will be transposed to be collected before operations begin; this is the block cipher signature.
     - Any possible transposition is necessarily a subset of an arbitrary substitution; thus, transposition can be seen as a particular keying subset of substitution.
     - Notice, however, that the usual avalanche signature of substitution is not present, and of course the actual data values are not changed at all by transposition, just moved about.
     - Also notice that we are close to using the idea of permutation in two very different ways: first as a particular n-bit to n-bit substitution, and second as a particular re-arrangement of characters in the block. These have wildly different ciphering effects.

B. STREAM CIPHER

- A stream cipher does *not* need to accumulate some amount of data or multiple data elements for ciphering to complete. (Since we define only two main "types" of cipher, a stream cipher is the opposite of a block cipher and vise versa. It is extremely important that the definitions for block and stream ciphering enclose the universe of all possible ciphers.)
- A stream cipher has the ability to transform individual elements one-by-one. The actual transformation usually is a block transformation, and may be repeated with the same or different keying.
- In a stream cipher, data diffusion may or may not occur, but if it does, it is necessarily one-way (from earlier to later elements).
- Since elements are ciphered one-by-one, changing part of the plaintext can affect that part and possibly *later* parts of the ciphertext; this is a stream cipher signature.
- The simple re-use of a block transformation to cover more data than a single block is a stream operation.

1. **CONFUSION SEQUENCE**
   - With a truly random sequence, used once, we have a one time pad.
   - With a pseudorandom confusion sequence and a simple additive combiner, we have a Vernam cipher.
   - A simple additive transformation becomes weak upon the second character ciphered, or immediately, under known plaintext, making strength dependent on the confusion sequence.
   - More complex transformations imply the need for correspondingly less strong confusion sequences.

   . **Autokey**
     - Normally the use of ciphertext, but also perhaps plaintext, as the cipher key.
     - Can create a random-like confusion stream which will re-synchronize after ciphertext data loss.
     - Under known-plaintext, the common "ciphertext feedback" version exposes both the confusion sequence and the input which creates that sequence. This is a lot of pressure on a single transformation.
2. **MONOALPHABETIC** (e.g., DES CBC)
   - The repeated use of a single fixed substitution.
   - A conventional block cipher *simulates* a large substitution.
   - A substitution becomes weak when its code values are re-used.
   - Code value re-use can be minimized by randomizing the plaintext block (e.g., CBC). This distributes the plaintext evenly across the possible block values, but at some point the transformation itself must change or be exposed.
   - Another alternative is to use a very large block so that code value re-use is made exceedingly unlikely. A large block also has room for a dynamic keying field which would make code value re-use even more unlikely.
3. **POLYALPHABETIC**
   - The use of multiple fixed substitutions.
   - By itself, the use of multiple alphabets in a regular sequence is inherently not much stronger than just a single alphabet.
   - It is of course possible to select an alphabet or transformation at pseudo-random, for example by re-keying DES after every block ciphered. This brings back sequence strength as an issue, and opens up the sequence generator starting state as an IV.
   - A related possibility is the use of a Latin square combiner which effectively selects among a balanced set of different fixed substitution alphabets.

   . **Cylinder**
     - A cipher which has or simulates the use of a number of different alphabet disks on a common rod.
     - Primary keying is the arrangement of the alphabet around each disk, and the selection and arrangement of disks.
     - By entering the plaintext on one row, any of n-1 other rows can be sent as ciphertext; this selection is an IV.
     - If the plaintext data are redundant, it is possible to avoid sending the IV by selecting the one of n-1 possible decipherings which shows redundancy. But this is not generally possible when ciphering arbitrary binary data.
     - If an IV is selected first, each character ciphering in that "chunk" is independent of each other ciphering. There is no data diffusion.
     - In general, each disk is used at fixed periodic intervals through the text, which is weak.
     - The ciphertext selection is homophonic, in the sense that different ciphertext rows each represent exactly the same plaintext.
     - Cylinder operation is **not** polyphonic in the usual sense: While a single ciphertext *can* imply any other row is plaintext, generally only one row has a reasonable plaintext meaning.
4. **DYNAMIC**
   - The use of one (monoalphabetic) or multiple (polyalphabetic) substitutions which *change* during ciphering.
5. **ITERATIVE**
   - The iterative re-use of a stream cipher with a new random IV on each iteration so as to eventually achieve the effect of a message key.
   - Each iteration seemingly must expand the ciphertext by the size of the IV, although this is probably about the same expansion we would have with a message key.
   - Unfortunately, each iteration will take some time.

Ciphering

The use of a cipher. The general term which includes both enciphering and deciphering.

Ciphertext

The result of enciphering. Ciphertext will contain the same information as the original plaintext, but hide the original information, typically under the control of a key. Without the key it should be impractical to recover the original information from the ciphertext.

Ciphertext Expansion

When the ciphertext is larger than the original plaintext.

Ciphertext expansion is the general situation: Stream ciphers need a message key, and block ciphers with a small block need some form of plaintext randomization, which generally needs an IV to protect the first block. Only block ciphers with a large size block generally can avoid ciphertext expansion, and then only if each block can be expected to hold sufficient uniqueness or "entropy" to prevent a codebook attack.

It is certainly true that in most situations of new construction a few extra bytes are not going to be a problem. However, in some situations, and especially when a cipher is to be installed into an existing system, the ability to encipher data *without* requiring additional storage can be a big advantage. Ciphering data without expansion supports the ciphering of data structures which have been defined and fixed by the rest of the system, provided only that one can place the cipher at the interface "between" two parts of the system. This is also especially efficient, as it avoids the process of acquiring a different, larger, amount of store for each ciphering. Such an installation also can apply to the entire system, and not require the re-engineering of all applications to support cryptography in each one.

Ciphony

Audio or voice encryption. A contraction of "ciphered telephony."

Circuit

The "circular" flow of electrons from a power source, through conductors and components and back to the power source. Or the arrangement of components which allows such flow and performs some function.

Clock

A repetitive or cyclic timing signal to coordinate state changes in a digital system. A clock can coordinate the movement of data and results through various stages of processing. Although a clock signal is digital, the source of the repetitive signal is almost always an analog circuit.

In an analog system we might produce a known delay by slowly charging a capacitor and measuring the voltage across it continuously until the voltage reaches the desired level. A big problem with this is that the circuit becomes increasingly susceptible to noise at the end of the interval.

In a digital system we create a delay by simply counting clock cycles. Since all external operations are digital, noise effects are virtually eliminated, and we can easily create accurate delays which are as long as the count in any counter we can build.

Closed

An operation on a set which produces only elements in that set.

Code

Symbols or values which stand for symbols, values, sequences, or even operations (as in computer "opcodes"). As opposed to a cipher, which operates only on individual characters or bits, classically, codes also represent words, phrases, and entire sentences. One application was to decrease the cost of telegraph messages. In modern usage, a code is often simply a correspondence between information (such as character symbols) and values (such as the ASCII code or Base-64), although computer opcodes do have independent meanings and variable lengths.

Coding is a very basic part of modern computation and generally implies no secrecy or information hiding. Some codes are "secret codes," however, and then the transformation between the information and the coding is kept secret. Also see: cryptography and substitution.

Codebook

Literally, the listing or "book" of code transformations. More generally, any collection of such transformations. Classically, letters, common words and useful phrases were numbered in a codebook; messages transformed into those numbers were "coded messages." Also see nomenclator. A "codebook style cipher" refers to a block cipher.

Codebook Attack

A form of attack in which The Opponent simply tries to build or collect a codebook of all the possible transformations between plaintext and ciphertext under a single key. This is the classic approach we normally think of as "codebreaking."

The usual ciphertext-only approach depends upon the plaintext having strong statistical biases which make some values far more probable than others, and also more probable in the context of particular preceding known values. Such attacks can be defeated if the plaintext data are randomized and thus evenly and independently distributed among the possible values. (This may have been the motivation for the use of a random confusion sequence in a stream cipher.)

When a codebook attack is possible on a block cipher, the complexity of the attack is controlled by the size of the block (that is, the number of elements in the codebook) and not the strength of the cipher. This means that a codebook attack would be equally effective against either DES or Triple-DES.

One way a block cipher can avoid a codebook attack is by having a large block size which will contain an unsearchable amount of plaintext "uniqueness" or entropy. Another approach is to randomize the plaintext block, often by using an operating mode such as CBC.

Combination

The mathematical term for any particular subset of symbols, independent of order. (Also called the binomial coefficient.) The number of combinations of *n* things, taken *k* at a time, read "*n* choose *k*" is:

```
 n
( ) = C(n,k) =  n! / (k! (n-k)!)
 k
```

See the combinations section of the Ciphers By Ritter / JavaScript computation pages. Also see permutation.

Combinatoric

Combinatorics is a branch of mathematics, like analysis or number theory. Combinatorics is often related to counting the subsets of finite sets. One result is to help us to understand the probability of a particular subset in the universe of possible values.

Consider a block cipher: For any given size block, there is some fixed number of possible messages. Since every enciphering must be reversible (deciphering must work), we have a 1:1 mapping between plaintext and ciphertext blocks. The set of all plaintext values and the set of all ciphertext values is the same set; particular values just have different meanings in each set.

Keying gives us no more ciphertext values, it only re-uses the values which are available. Thus, keying a block cipher consists of selecting a particular arrangement or permutation of the possible block values. Permutations are a combinatoric topic. Using combinatorics we can talk about the number of possible permutations or keys in a block cipher, or in cipher components like substitution tables.

Permutations can be thought of as the number of unique arrangements of a given length on a particular set. Other combinatoric concepts include binomials and combinations (the number of unique given-length subsets of a given set).

Combiner

In a cryptographic context, a combiner is a mechanism which mixes two data sources into a single result. A "combiner style cipher" refers to a stream cipher.

Reversible combiners are used to encipher plaintext into ciphertext in a stream cipher. The ciphertext is then deciphered into plaintext using a related inverse or extractor mechanism.

Irreversible or non-invertible combiners are often used to mix multiple RNG's into a single confusion sequence, also for use in stream cipher designs.

Also see balanced combiner, additive combiner and complete, and The Story of Combiner Correlation: A Literature Survey, in the Literature Surveys and Reviews section of the Ciphers By Ritter page.

Commutative

A dyadic operation in which exchanging the two argument values must produce the same result: a + b = b + a.

Also see: associative and distributive.

Complete

A term used in S-box analysis to describe a property of the value arrangement in an invertible substitution or, equivalently, a block cipher. If we have some input value, and then change one bit in

that value, we expect about half the output bits to change; this is the result of diffusion; when partial diffusion is repeated we develop avalanche; and the ultimate result is strict avalanche. *Completeness* tightens this concept and requires that changing a particular input bit produce a change in a particular output bit, at some point in the transformation (that is, for at least one input value). Completeness requires that this relationship occur at least once for *every* combination of input bit and output bit. It is tempting to generalize the definition to apply to multi-bit element values, where this makes more sense.

Completeness does *not* require that an input bit change an output bit for *every* input value (which would not make sense anyway, since *every* output bit must be changed at *some* point, and if they all had to change at *every* point, we would have *all* the output bits changing, instead of the desired half). The inverse of a complete function is not necessarily also complete.

As originally defined in Kam and Davida:

> "For every possible key value, every output bit c_i of the SP network depends upon all input bits $p_1,...,p_n$ and not just a proper subset of the input bits." [p.748]

Kam, J. and G. Davida. 1979. Structured Design of Substitution-Permutation Encryption Networks. *IEEE Transactions on Computers.* C-28(10): 747-753.

Component

A part of a larger construction; a building-block in an overall design or system. Modern digital design is based on the use of a few general classes of pre-defined, fully-specified parts. Since even digital logic can use or even require analog values internally, by enclosing these values the logic component can hide complexity and present the appearance of a fully digital device.

The most successful components are extremely general and can be used in many different ways. Even as a brick is independent of the infinite variety of brick buildings, a flip-flop is independent of the infinite variety of logic machines which use flip-flops.

The source of the ability to design and build a wide variety of different electronic logic machines is the ability to interconnect and use a few very basic but very general parts.

Electronic components include

- passive components like resistors, capacitors, and inductors;
- active components like transistors and even relays, and
- whole varieties of active electronic logic devices, including flip-flops, shift registers, and state storage, or memory.

Cryptographic system components include:

- Nonlinear transformations, such as S-boxes / substitution tables,
- key hashing, such as CRC,
- random number generators, such as additive RNG's,
- sequence isolators such as jitterizers,
- combiners, such as Dynamic Substitution, Latin squares, and exclusive-OR,
- mixers, such as Balanced Block Mixers, or orthogonal Latin squares.

Computer

Originally the job title for a person who performed a laborious sequence of arithmetic computations. Now a machine for performing such calculations.

A logic machine with:

1. Some limited set of fundamental computations. Typical operations include simple arithmetic and Boolean logic. Each operation is selected by a particular operation code value or "opcode." This is a hardware interpretation of the opcode.
2. The ability to follow a list of instructions or commands, performing each in sequence. Thus capable of simulating a wide variety of far more complex "instructions."
3. The ability to execute or perform at least some instructions conditionally, based on parameter values or intermediate results.
4. The ability to store values into a numbered "address space" which is far larger than the instruction set, and later to recover those values when desired.

Also see: source code, object code and software.

Conductor

A material in which electron flow occurs easily. Typically a metal; usually copper, sometimes silver, brass or even aluminum. A wire. As opposed to an insulator.

Confusion

Those parts of a cipher mechanism which change the correspondence between input values and output values. In contrast to diffusion.

Confusion Sequence

The sequence combined with data in a stream cipher. Normally produced by a random number generator, it is also called a "running key."

Contextual

In the study of logic, an observed fact dependent upon other facts *not* being observed. Or a statement which is conditionally true, provided other unmentioned conditions have the appropriate state. As opposed to absolute.

Conventional Cipher

A secret key cipher.

Congruence

Casually speaking, the remainder after a division of integers.

In number theory we say than integer a (exactly) *divides* integer b (denoted a | b) if and only if there is an integer k such that ak = b.

In number theory we say that integer a is *congruent* to integer b *modulo* m, denoted a = b (mod m), if and only if m | (a - b). Here m is the divisor or *modulus.*

Convolution

Polynomial multiplication. A multiplication of each term against each other term, with no "carries" from term to term. Also see correlation.

Used in the analysis of signal processing to develop the response of a processing system to a complicated real-valued input signal. The input signal is first separated into some number of discrete impulses. Then the system response to an impulse -- the output level at each unit time delay after the impulse -- is determined. Finally, the expected response is computed as the sum of the contributions from each input impulse, multiplied by the magnitude of each impulse. This is an approximation to the convolution integral with an infinite number of infinitesimal delays. Although originally accomplished graphically, the process is just polynomial multiplication.

It is apparently possible to compute the convolution of two sequences by taking the FFT of each, multiplying these results term-by-term, then taking the inverse FFT. While there is an analogous relationship in the FWT, in this case the "delays" between the sequences represent mod 2 distance differences, which may or may not be useful.

Correlation

In general, the probability that two sequences of symbols will, in any position, have the same symbol. We expect two random binary sequences to have the same symbols about half the time.

One way to evaluate the correlation of two real-valued sequences is to multiply them together term-by-term and sum all results. If we do this for all possible "delays" between the two sequences, we get a "vector" or 1-dimensional array of correlations which is a convolution. Then the maximum value represents the delay with the best correlation.

Correlation Coefficient

The value from -1 to +1 describing the correlation of two binary sequences, averaged over the length of interest. Correlation coefficient values are related to the probability that, given a symbol from one sequence, the other sequence will have that same symbol. A value of:

- -1 implies a 0.0 probability (the second sequence is the complement of the first),
- 0 implies a 0.5 probability (the sequences are uncorrelated), and
- +1 implies a 1.0 probability (the sequences are the same).

"The correlation coefficient associated with a pair of Boolean functions *f(a)* and *g(a)* is denoted by C(f,g) and is given by

```
C(f,g) = 2 * prob(f(a) = g(a)) - 1 ."
```

Daemen, J., R. Govaerts and J. Vanderwalle. 1994. Correlation Matrices. *Fast Software Encryption.* 276. Springer-Verlag.

CRC

Cyclic Redundancy Check: A fast error-check hash based on mod 2 polynomial operations.

A CRC is essentially a fast remainder operation over a huge numeric value which is the data. (For best speed, the actual computation occurs as mod 2 polynomial operations.) The CRC result is an excellent (but linear) hash value corresponding to the data.

No CRC has any appreciable strength, but some applications -- even in cryptography -- *need* no strength:

- One example is authentication, provided the linear CRC hash result is protected by a block cipher.
- Another example is key processing, where the uncertainty in a User Key phrase of arbitrary size is collected into a hash result of fixed size. In general, the hash result would be just as good for The Opponent as the original key phrase, so no strength shield could possibly improve the situation.
- A third example is the accumulation of the uncertainty in slightly uncertain physically random events. When true randomness is accumulated, it is already as unknowable as any strength shield could make it.

Cryptanalysis

That aspect of cryptology which concerns the strength analysis of a cryptographic system, and the penetration or breaking of a cryptographic system. Also "codebreaking."

Because there is no theory which guarantees strength for any conventional cipher, ciphers traditionally have been considered "strong" when they have been used for a long time with "nobody" knowing how to break them easily. Cryptanalysis seeks to improve this process by applying the known attack strategies to new ciphers, and by actively seeking new ones. It is normal to assume that at least known-plaintext is available; often, defined-plaintext is assumed. The result is typically some value for the amount of "work" which will achieve a "break" (even if that value is impractical); this is "the" strength of the cipher.

But while cryptanalysis *can* prove "weakness" for a given level of effort, cryptanalysis *cannot* prove that there is no simpler attack:

Lack of proof of weakness is not proof of strength.

Indeed, when ciphers are used for real, The Opponents can hardly be expected to advertise a successful break, but will instead work hard to reassure users that their ciphers are still secure. The fact that *apparently* "nobody" knows how to break a cipher is somewhat less reassuring from this viewpoint. In this context, using a wide variety of different ciphers can make good sense: This reduces the value of the information protected by any particular cipher, which thus reduces the rewards from even a successful attack. Having a numerous ciphers also requires The Opponents to field far greater resources to identify, analyze, and automate breaking (when possible) of each different cipher.

Many academic attacks are essentially theoretical, involving huge amounts of data and computation. But even when a direct technical attack is *practical*, that may be the most difficult, expensive and time-consuming way to obtain the desired information. Other methods include making a paper copy, stealing a copy, bribery, coercion, and electromagnetic monitoring. No cipher can keep secret something which has been otherwise revealed. Information security thus involves far more than just cryptography, and even a cryptographic system is more than just a cipher. Even finding that information has been revealed does not mean that a cipher has been broken.

At one time it was reasonable to say: "Any cipher a man can make, another man can break." However, with the advent of serious computer-based cryptography, that statement is no longer valid, *provided* that every detail is properly handled. This, of course, often turns out to not be the case.

Cryptanalyst

Someone who attacks ciphers with cryptanalysis. A "codebreaker." Often called the Opponent by cryptographers, in recognition of the (serious) game of thrust and parry between these parties.

Cryptographer

Someone who creates ciphers using cryptography.

Cryptographic Mechanism

A process for enciphering and/or deciphering, or an implementation (for example, hardware, computer software, hybrid, or the like) for performing that process. See also cryptography and mechanism.

Cryptography

Greek for "hidden writing." The art and science of transforming information into an intermediate form which secures that information while in storage or in transit. A part of cryptology, further divided into secret codes and ciphers. As opposed to steganography, which seeks to hide the existence of any message, cryptography seeks to render a message unintelligible *even when the message is completely exposed*.

Cryptography includes at least:

- secrecy (*confidentiality*, or *privacy*, or *information security*) and
- message authentication (*integrity*).

Cryptography may also include:

- *nonrepudiation* (the inability to deny sending a message),
- *access control* (*user* or *source* authentication), and
- *availability* (keeping security services available).

Modern cryptography generally depends upon translating a message into one of an astronomical number of different intermediate representations, or ciphertexts, as selected by a key. If all possible intermediate representations have similar appearance, it may be necessary to try all possible keys to find the one which deciphers the message. By creating mechanisms with an astronomical number of keys, we can make this approach impractical.

Cryptography may also be seen as a zero-sum game, where a cryptographer competes against a cryptanalyst. We might call this the cryptography war.

Cryptography War

Cryptography may be seen as a dynamic *battle* between cryptographer and cryptanalyst. The cryptographer tries to produce a cipher which can retain secrecy. Then, when it becomes worthwhile, one or more cryptanalysts try to penetrate that secrecy by attacking the cipher. Fortunately for the war, even after fifty years of mathematical cryptology, not *one* practical cipher has been accepted as *proven* secure in practice. (See, for example, the one-time pad.)

Note that the successful cryptanalyst must keep good attacks secret, or the opposing cryptographer will just produce a stronger cipher. This means that the cryptographer is in the odd position of never knowing whether his or her best cipher designs are successful, or which side is winning.

Cryptographers are often scientists who are trained to ignore unsubstantiated claims. But there will *be* no substantiation when a cipher system is attacked and broken for real, yet continued use will endanger all messages so "protected." Thus, it is a very reasonable policy to not adopt a widely-used cipher, and to change ciphers periodically.

Cryptology

The field of study which generally includes steganography, cryptography and cryptanalysis.

Current

The measure of electron flow, in amperes. Current is analogous to the amount of water *flow*, as opposed to *pressure* or voltage. A flowing electrical current will create a magnetic field around the conductor. A changing electrical current may create an electromagnetic field.

---

dB

decibel.

DC

Direct Current: Electrical power which flows in one direction, more or less constantly. As opposed to AC.

Most electronic devices require DC -- at least internally -- for proper operation, so a substantial part of modern design is the "power supply" which converts 120 VAC wall power into 12 VDC, 5 VDC and/or 3 VDC as needed by the circuit and active devices.

Debug

The interactive analytical process of correcting the design of a complex system. A normal part of the development process, although when bugs are not caught during development, they can remain in production systems.

Contrary to naive expectations, a complex system almost never performs as desired when first realized. Both hardware and software system design environments generally deal with systems which are not working. (When a system *really* works, the design and development process is generally over.) Debugging involves identifying problems, analyzing the source of those problems, then changing the construction to fix the problem. (Hopefully, the fix will not itself create new problems.) This form of interactive analysis can be especially difficult because the realized design may not actually be what is described in the schematics, flow-charts, or other working documents: To some extent the real system is unknown.

When a system has many problems, the problems tend to interact, which can make the identification of a particular cause very difficult. This can be managed by "shrinking" the system: first by partitioning the design into components and testing those components, and then by temporarily disabling or removing sections so as to identify the section in which the problem lies. Eventually, with enough testing, partitioning and analysis, the source of any problem can be identified. Some "problems," however, turn out to be the unexpected implications of a complex design and are sometimes accepted as "features" rather than the alternative of a complete design overhaul.

Decipher

The process which can reveal the information or plaintext hidden in message ciphertext (provided it is the correct process, with the proper key). The inverse of encipher.

Decryption

The general term for extracting information which was hidden by encryption.

Deductive Reasoning

In the study of logic, reasoning about a particular case from one or more general statements; a proof. Also see: inductive reasoning and fallacy.

Defined Plaintext Attack

A form of attack in which the Opponent can present arbitrary plaintext to be enciphered, and then capture the resulting ciphertext. The ultimate form of known plaintext attack.

A defined plaintext attack can be a problem for systems which allow unauthorized users to present arbitrary messages for ciphering. Such attack can be made difficult by allowing only authorized users to encipher data, by allowing only a few messages to be enciphered between key changes, by changing keys frequently, and by enciphering each message in a different random message key.

Degrees of Freedom

In statistics, the number of completely independent values in a sample. The number of sampled values or observations or bins, less the number of defined or freedom-limiting relationships or "constraints" between those values.

If we choose two values completely independently, we have a DF of 2. But if we must choose two values such that the second is twice the first, we can choose only the first value independently. Imposing a relationship on one of the sampled value means that we will have a DF of one less than the number of samples, even though we may end up with apparently similar sample values.

In a typical goodness of fit test such as chi-square, the reference distribution (the expected counts) is normalized to give the same number of counts as the experiment. This is a constraint, so if we have N bins, we will have a DF of N - 1.

DES

The particular block cipher which is the U.S. Data Encryption Standard. A 64-bit block cipher with a 56-bit key organized as 16 rounds of operations.

Decibel

Ten times the base-10 logarithm of the ratio of two power values. Denoted by dB. One-tenth of a bel.

When voltages or currents are measured, power changes as the square of these values, so a decibel is twenty times the base-10 logarithm of the ratio of two voltages or currents.

Decimal

Base 10: The numerical representation in which each digit has an alphabet of ten symbols, usually 0 through 9. Also see: binary, octal, and hexadecimal.

Design Strength

The keyspace; the effort required for a brute force attack.

Deterministic

A process whose sequence of operations is fully determined by its initial state. A mechanical or clockwork-like process whose outcome is inevitable, given its initial setting. Pseudorandom.

Dictionary Attack

Typically an attack on a secret password. A dictionary of common passwords is developed, and a brute force attack conducted on the target with each common password.

Differential Cryptanalysis

A form of attack in which the difference between values (or keys) is used to gain some information about the system.

Also see Differential Cryptanalysis: A Literature Survey, in the Literature Surveys and Reviews section of the Ciphers By Ritter page.

Diffusion

Diffusion is the property of an operation such that changing one bit (or byte) of the input will change adjacent or near-by bits (or bytes) after the operation. In a block cipher, diffusion propagates bit-changes from one part of a block to other parts of the block. Diffusion requires mixing, and the step-by-step process of increasing diffusion is described as avalanche. Diffusion is in contrast to confusion.

Normally we speak of *data* diffusion, in which changing a tiny part of the plaintext data may affect the whole ciphertext. But we can also speak of *key* diffusion, in which changing even a tiny part of the key should change each bit in the ciphertext with probability 0.5.

Perhaps the best diffusing component is substitution, but this diffuses only within a single substituted value. Substitution-permutation ciphers get around this by moving the bits of each substituted element to other elements, substituting again, and repeating. But this only provides guaranteed diffusion if particular substitution tables are constructed. Another alternative is to use some sort of Balanced Block Mixing which has an inherently guaranteed diffusion, or a Variable Size Block Cipher construction. Also see Overall Diffusion.

Digital

Pertaining to discrete or distinct finite values. As opposed to analog or continuous quantities.

Diode

An electronic device with two terminals which allows current to flow in only one direction.

Distribution

In statistics, the range of values which a random variable, and the probability that each value or range of values will occur. Also the probability of test statistic values for the case "nothing unusual found," which is the null hypothesis.

If we have a *discrete* distribution, with a finite number of possible result values, we can speak of "frequency" and "probability" distributions: The "frequency distribution" is the expected *number* of occurrences for each possible value, in a particular sample size. The "probability distribution" is the *probability* of getting each value, normalized to a probability of 1.0 over the sum of all possible values.

Here is a graph of a typical "discrete probability distribution" or "discrete probability density function," which displays the probability of getting a particular statistic value for the case "nothing unusual found":

```
0.1|        ***
   |       *   *          Y = Probability of X
 Y |     **     **        y = P(x)
   | ****         ****
0.0 ------------------
             X
```

Unfortunately, it is not really possible to think in the same way about continuous distributions: Since continuous distributions have an infinite number of possible values, the probability of getting any *particular* value is zero. For continuous distributions, we instead talk about the probability of getting a value in some subrange of the overall distribution. We are often concerned with the probability of getting a particular value or below, or the probability of a particular value or above.

Here is a graph of the related "cumulative probability distribution" or "cumulative distribution function" (c.d.f.) for the case "nothing unusual found":

```
1.0|            ******
   |          **          Y = Probability (0.0 to 1.0) of finding
 Y |         *                a value which is x or less
   |       **
0.0 -******-----------
             X
```

The c.d.f. is just the sum of all probabilities for a given value or less. This is the usual sort of function used to interpret a statistic: Given some result, we can look up the probability of a lesser value (normally called *p*) or a greater value (called *q* = 1.0 - *p*).

Usually, a test statistic is designed so that extreme values are not likely to occur by chance in the case "nothing unusual found" which is the null hypothesis. So if we *do* find extreme values, we have a strong argument that the results were not due simply to random sampling or other random effects, and may choose to reject the null hypothesis and thus accept the alternative hypothesis.

Common discrete distributions include the binomial distribution, and the Poisson distribution.

Distributive

The case of a dyadic operation, which may be called "multiplication," which can be applied to equations involving another dyadic operation, which may be called "addition," such that: a(b + c) = ab + ac and (b + c)a = ba + bc.

Also see: associative and commutative.

Divide and Conquer

The general concept of being able to split a complexity into several parts, each part naturally being less complex than the total. If this is possible, The Opponent may be able to solve all of the parts far easier than the supposedly complex whole. Often part of an attack.

This is a particular danger in cryptosystems, since most ciphers are built from less-complex parts. Indeed, a major role of cryptographic design is to combine small component parts into a larger complex system which cannot be split apart.

Domain

The set of all arguments *x* which can be applied to a mapping. Also see range.

Dyadic

Relating to *dyad*, which is Greek for dual or having two parts. In particular, a function with two inputs or arguments. Also see: monadic, unary and binary.

Dynamic Keying

That aspect of a cipher which allows a key to be changed with minimal overhead. A dynamically-keyed block cipher might impose little or no additional computation to change a key on a block-by-block basis. The dynamic aspect of keying could be just one of multiple keying mechanisms in the same cipher.

One way to have a dynamic key in a block cipher is to include the key value along with the plaintext data. But this is normally practical only with blocks of huge size, or variable size blocks.

Another way to have a dynamic key in a block cipher is to add a confusion layer which mixes the key value with the block. For example, exclusive-OR could be used to mix a 64-bit key with a 64-bit data block.

Dynamic Substitution Combiner

The combining mechanism described in U.S. Patent 4,979,832 (see the Dynamic Substitution articles on the Ciphers By Ritter page).

Dynamic Substitution is the use of an invertible substitution table in which the arrangement of the entries changes dynamically during operation. This is particularly useful as a strong replacement for the strengthless exclusive-OR combiner in stream ciphers.

The arrangement of a keyed table starts out unknown to an Opponent. From the Opponent's point of view, each table entry could be any possible value with uniform probability. But after the first value is mapped through that table, the used transformation (table entry) is at least potentially exposed, and no longer can be considered a completely unknown probability. Dynamic Substitution acts to make the used transformation again completely unknown and unbiased, by allowing it to take on any possible mapping. As a first approximation, the amount of information leaked about table contents is replaced by information used to re-define each used entry.

In the usual case, an invertible substitution table is keyed by shuffling under the control of a random number generator. One combiner input value is used to select a value from within that table to be the result or output. The other combiner input value is used simply to select an entry, and then the values at the two selected entries are exchanged. So as soon as a plaintext mapping is used, it is immediately reset to any possibility, and the more often any plaintext value occurs, the more often that transformation changes.

Also see Balanced Block Mixing, and Variable Size Block Cipher.

Dynamic Transposition

A block cipher which first creates an exact bit-balance within each block, and then shuffles the bits within a block, each block being permuted independently from a keyed random number generator.

Since each block -- plaintext or ciphertext -- contains exactly the same number of 1's and 0's, every possible plaintext block is just some permutation of any possible ciphertext block. And since any possible plaintext block can be produced from any ciphertext block in a vast plethora of different ways, the keying sequence is hidden even from known plaintext. And defined plaintext is easily defeated with the usual message key. To the extent that every possible plaintext block can be produced, the cipher approaches perfect secrecy.

See the article Transposition Cipher with Pseudo-Random Shuffling: The Dynamic Transposition Combiner.

ECB

ECB or Electronic Code Book is an operating mode for block ciphers. Presumably the name comes from the observation that a block cipher under a fixed key functions much like a physical codebook: Each possible plaintext block value has a corresponding ciphertext value, and vise versa.

ECB is the naive method of applying a block cipher, in that the plaintext is simply partitioned into appropriate size blocks, and each block is enciphered separately and independently. When we have a small block size, ECB is generally unwise, because language text has biased statistics which will result in some block values being re-used frequently, and this repetition will show up in the raw ciphertext. This is the basis for a successful codebook attack.

On the other hand, if we have a large block, we may expect it to contain enough (at least, say, 64 bits) uniqueness or "entropy" to prevent a codebook attack. In that case, ECB mode has the advantage of supporting independent ciphering of each block. This, in turn, supports various things, like the use of multiple ciphering hardware operating in parallel for higher speeds.

As another example, modern packet-switching network technologies often deliver raw packets out of order. The packets will be re-ordered eventually, but having out-of-sequence packets can be a problem for low-level ciphering if the blocks are not ciphered independently.

Electric Field

The fundamental physical force resulting from the attraction of opposing charges.

Electromagnetic Field

The remarkable self-propagating physical field consisting of energy distributed between electric and magnetic fields. Energy in the electric or potential field collapses and creates or "charges up" a magnetic field. Energy in the magnetic field collapses and "charges up" an electric field. This process allows physical electrical and magnetic fields -- two fairly short-range phenomena -- to "propagate" and thus carry energy over relatively large distances at the speed of light. Examples include light, "radio" waves (including TV, cell phones, etc.), and microwave cooking.

It is important to distinguish between a true electromagnetic field, and the simpler and range-limited electric and magnetic fields produced by an electrical clock, motor, or power lines. It is also important to distinguish between the light-like expanding or "radiating" property of an electromagnetic field, and the damaging ionizing radiation produced by a radioactive source.

As far as we know -- and a great many experiments have been conducted on this -- electromagnetic waves are not life-threatening (unless they transfer enough power to dangerously heat the water in our cells). The belief that electromagnetic fields are not dangerous is also *reasonable*, since light itself is an electromagnetic wave, and life on Earth developed in the context of the electromagnetic field from the Sun. Indeed, plants actually use that field to their and our great benefit.

Electronic

Having to do with the control and use of physical electrons, as electrical potential or voltage, electrical flow or current, and generally both. See hardware and component.

Encipher

The process which will transform information or plaintext into one of plethora of intermediate forms or ciphertext, as selected by a key. The inverse of decipher.

Encryption

The general term for hiding information in secret code or cipher.

Entropy

In information theory, our "uncertainty" as to the value of a random variable. Given the non-zero probability (*p*) of each value (*i*), we can calculate an entropy (*H*) in bits for random variable *X* as:

$$H(X) = -SUM(p_i \log2 p_i)$$

Although entropy is sometimes taken as a measure of randomness, calculating entropy requires a knowledge of the probabilities of each value which we often can attain only by sampling. This means that we do not *really* know the "true" probabilities, but only those we see in our samples. And the "true" probabilities may change through time.

By itself, calculated entropy also does not detect any underlying order that might exist between value probabilities, such as a correlation, or a linear relationship, or any other aspect of cryptographically-weak randomness. The "true entropy" of a random number generator is just the number of bits in the state of that generator, as opposed to an entropy computation on the sequence it produces. So a high entropy value does *not* imply that a really-random source really *is* random, or indeed have any relationship to the amount of cryptographic randomness present.

Ergodic

In statistics and information theory, a particularly "simple" and easily modelled stationary (homogenous) stochastic (random) process (function) in which the "temporal average" is the same as the "ensemble average." In general, a process in which no state is prevented from re-occurring. Ergodic processes are the basis for many important results in information theory, and are thus a technical requirement before those results can be applied.

Here we have all three possible sequences from a **non**-ergodic process: **across** we have the average of symbols through time (the "temporal average"), and **down** we have the average of symbols in a particular position over all possible sequences (the "ensemble average"):

```
A B A B A B ...   p(A) = 0.5, p(B) = 0.5, p(E) = 0.0
B A B A B A ...   p(A) = 0.5, p(B) = 0.5, p(E) = 0.0
E E E E E E ...   p(A) = 0.0, p(B) = 0.0, p(E) = 1.0
^ ^ ^ ^ ^ ^
+-+-+-+-+-+----   p(A) = 0.3, p(B) = 0.3, p(E) = 0.3
```

(From: Pierce, J. 1961. *Symbols, Signals and Noise.* Ch. 3)

When a process is non-ergodic, the measurements we take over time from one or a few sequences may not represent all the sequences which may be encountered.

Extractor

In a cryptographic context, an extractor is a mechanism which produces the inverse effect of a combiner. This allows data to be enciphered in a combiner, and then deciphered in an extractor. Sometimes an extractor is exactly the same as the combiner, as is the case for exclusive-OR.

Exclusive-OR

A Boolean logic function which is also mod 2 addition. Also called XOR.

Factorial

The *factorial* of natural number *n*, written **n!**, is the product of all integers from 1 to *n*.

See the factorials section of the Ciphers By Ritter / JavaScript computation pages.

Fallacy

In the philosophical study of logic, apparently-reasonable arguments which lead to false conclusions. Also see: inductive reasoning and deductive reasoning. Including:

. Fallacies of Insufficient Evidence
  1. Accident -- a special circumstance makes a rule inapplicable
  2. Hasty Generalization
  3. *non causa pro causa* ("False Cause")
     - *post hoc ergo propter hoc* ("after this therefore because of this")
     - *reductio ad absurdum* -- the assumption that a particular one of multiple assumptions is necessarily false if the argument leads to a contradiction.
  4. *ad ignorantium* ("Appeal to Ignorance") -- a belief which is assumed true because it is not proven false.
  5. Card Stacking -- a deliberate withholding of evidence which does not support the author's conclusions.

B. Fallacies of Irrelevance (*ignoratio elenchi*) -- ignoring the question
  1. *ad hominem* ("Name Calling").
  2. *ad populum* ("Plain Folks") -- an appeal to the prejudices and biases of the audience.
  3. *ad misericordiam* ("Appeal to Pity")
  4. *ad verecundiam* ("Inappropriate Authority") -- a testimonial from someone with expertise in a different field.
  5. *tu quoque* ("You Did It Too").
  6. *ad baculum* ("Appeal to force") -- e.g., threats.
  7. Red Herring -- information used to throw the discussion off track.
  8. Opposition ("Guilt by Association") -- to condemn an idea because of who is for it.
  9. Genetic -- attacking the source of the idea, rather than the idea itself.
  10. Bandwagon

C. Fallacies of Ambiguity
  1. Equivocation -- the use of a word in a sense different than that understood by the reader.
  2. Amphiboly -- some sentences admit more than one interpretation.
  3. Accent -- some sentences have different meanings depending on which word is stressed.
  4. Composition -- the implication that what is true of the parts must also be true of the whole.
  5. Division -- the implication that what is true of the whole must be true of its parts.
  6. False Analogy

D. Fallacies of the Misuse of Logic
  1. *petitio principii* ("Begging the Question") -- restating one of the premises as the conclusion; assuming the truth of a proposition which needs to be proven.
     - *circulus in probando* ("Circular Argument")
  2. *non sequitur* ("Does Not Follow") -- the stated conclusion does not follow from the evidence supplied.
  3. *plurimum interrogationum* ("Complex Question") -- e.g., "When did you stop beating your wife?"
  4. Garbled Syllogism -- an illogical argument phrased in logical terms.
  5. Either-Or -- assuming a question has only two sides.

Fast Walsh Transform

(Also Walsh-Hadamard transform.) When applied to a Boolean function, a Fast Walsh Transform is essentially a correlation count between the given function and each Walsh function. Since the Walsh functions are essentially the affine Boolean functions, the FWT computes the unexpected distance from a given function to each affine function. It does this in time proportional to *n* log n, for functions of *n* bits, with *n* some power of 2.

If two Boolean functions are *not* correlated, we expect them to agree half the time, which we might call the "expected distance." When two Boolean functions *are* correlated, they will have a distance greater or less than the expected distance, and we might call this difference the unexpected distance or UD. The UD can be positive or negative, representing distance to a particular affine function or its complement.

It is easy to do a Fast Walsh Transform by hand. (Well, I say "easy," then always struggle when I actually do it.) Let's do the FWT of function f: (1 0 0 1 1 1 0 0): First note that f has a binary power length, as required. Next, each pair of elements is modified by an "in-place butterfly"; that is, the values in each pair produce two results which replace the original pair, wherever they were originally located. The left result will be the two values added; the right will be the first less the second. That is,

```
(a',b') = (a+b, a-b)
```

So for the values (1,0), we get (1+0, 1-0) which is just (1,1). We start out pairing adjacent elements, then every other element, then every 4th element, and so on until the correct pairing is impossible, as shown:

```
original      1   0   0   1   1   1   0   0
              ^---^   ^---^   ^---^   ^---^

   first      1   1   1  -1   2   0   0   0
              ^-------^       ^-------^
                  ^-------^       ^-------^

  second      2   0   0   2   2   0   2   0
              ^---------------^
                  ^---------------^
                      ^---------------^
                          ^---------------^

      final      4   0   2   2   0   0  -2   2
```

The result is the "unexpected distance" to each affine Boolean function. The higher the absolute value, the greater the "linearity"; if we want the *non*linearity, we must subtract the absolute value of each unexpected distance from the expected value, which is half the number of bits in the function. Note that the range of possible values increases by a factor of 2 (in both positive and negative directions) in each sublayer mixing; this is information expansion, which we often try to avoid in cryptography.

Also see: Walsh-Hadamard Transforms: A Literature Survey, in the Literature Surveys and Reviews section of the Ciphers By Ritter page, and the Active Boolean Function Nonlinearity Measurement in JavaScript page of the Ciphers By Ritter / JavaScript computation pages.

The FWT provides a strong mathematical basis for block cipher mixing such that all input values will have an equal chance to affect all output values. Cryptographic mixing then occurs in butterfly operations based on balanced block mixing structures which replace the simple add / subtract butterfly in the FWT and confine the value ranges so information expansion does not occur. A related concept is the well-known FFT, which can use exactly the same mixing patterns as the FWT.

FCSR

Feedback with Carry Shift Register. A sequence generator analogous to a LFSR, but separately storing and using a "carry" value from the computation.

Feistel Construction

The Feistel construction is the widely-known method of constructing block ciphers used in DES. Horst Feistel worked for IBM in the 60's and 70's, and was awarded a number of crypto patents, including: 3,768,359, 3,768,360, and 4,316,055.

Normally, in a Feistel construction, the input block is split into two parts, one of which drives a transformation whose result is exclusive-OR combined into the other block. Then the "other block" value feeds the same transformation, whose result is exclusive-OR combined into the first block. This constitutes 2 of perhaps 16 "rounds."

```
L          R
|          |
|--> F --> +    round 1
|          |
+ <-- F <--|    round 2
|          |
v          v
L'         R'
```

One advantage of the Feistel construction is that the transformation does not need to be invertible. To reverse any particular layer, it is only necessary to apply the same transformation again, which will undo the changes of the original exclusive-OR.

A disadvantage of the Feistel construction is that diffusion depends upon the internal transformation. There is no guarantee of overall diffusion, and the number of rounds required is often found by experiment.

Fenced DES

A block cipher with three layers, in which the outer layers consist of fencing tables, and the inner layer consists of DES used as a component. For block widths over 64 bits, Balanced Block Mixing technology assures that any bit change is propagated to each DES operation.

Also see the Fenced DES section of the Ciphers By Ritter page, and A Keyed Shuffling System for Block Cipher Cryptography.

Fencing

Fencing is a term-of-art which describes a layer of substitution tables. In schematic or data-flow diagrams, the row of tiny substitution boxes stands like a picket fence between the data on each side.

Fencing Layer

A fencing layer is a variable size block cipher layer composed of small (and therefore realizable) substitutions. Typically the layer contains many separate keyed substitution tables. To make the layer extensible, either the substitutions can be re-used in some order, or in some pre-determined sequence, or the table to be used at each position selected by some computed value.

Fencing layers are also used in other types of cipher.

FFT

Fast Fourier Transform. A numerically advantageous way of computing a Fourier transform. Basically a way of transforming information from amplitude values sampled periodically through time, into amplitude values sampled periodically through complex frequency. The FFT performs this transformation in time proportional to n log n, for some n a power of 2.

While exceedingly valuable, the FFT tends to run into practical problems in use which can require a deep understanding of the process. For example, the transform assumes that the waveform is "stationary" and thus repetitive and continuous, which is rarely the case. As another example, sampling a continuous wave can create spurious "frequency" values related to the sampling and not the wave itself. Also the range of possible values increases by a factor of 2 (in both positive and negative directions) in every sublayer mixing; this is information expansion, which we often try to avoid in cryptography.

The FFT provides a strong mathematical basis for block cipher mixing such that all input values will have an equal chance to affect all output values. Cryptographic mixing then occurs in butterfly operations based on balanced block mixing structures which replace the simple add / subtract butterfly in the FFT and confine the value ranges so information expansion does not occur. A related concept is the fast Walsh-Hadamard transform (FWT), which can use exactly the same mixing patterns as the FFT.

Field

In abstract algebra, a commutative ring in which all non-zero elements have a multiplicative inverse. (This means we can divide.)

In general, a field supports the four basic operations (addition, subtraction, multiplication and division), and satisfies the normal rules of arithmetic. An operation on any two elements in a field is a result which is also an element in the field.

Examples of fields include rings of integers modulo some prime. Here are multiplication tables under mod 2, mod 3 and mod 4:

```
     0  1          0  1  2          0  1  2  3

0    0  0     0    0  0  0     0    0  0  0  0
1    0  1     1    0  1  2     1    0  1  2  3
              2    0  2  1     2    0  2  0  2
                               3    0  3  2  1
```

In a field, each element must have an inverse, and the product of an element and its inverse is 1. This means that every non-zero row and column of the multiplication table for a field must contain a 1. Since row 2 of the mod 4 table does not contain a 1, the set of integers mod 4 is not a field.

The *order* of a field is the number of elements in that field. The integers mod *p* form a finite field of order *p*. Similarly, mod 2 polynomials will form a field with respect to an irreducible polynomial, and will have order 2^n, which is a very useful size.

Finite Field

A Galois field: A mathematical field of non-infinite order. As opposed to an *infinite field*, such as the integers, rationals, reals and complex numbers.

- In a finite field, every nonzero element *x* can be squared, cubed, and so on, and at some power will eventually become 1. The smallest (positive) power *n* at which $x^n = 1$ is the *order* of element *x*. This of course makes *x* an "*nth root of unity*," in that it satisfies the equation $x^n = 1$.
- A finite field of order *q* will have one or more *primitive* elements *a* whose order is *q*-1 and whose powers cover all nonzero field elements.
- For every element *x* in a finite field of order *q*, $x^q = x$.

Flip-Flop

A class of digital logic component which has a single bit of state with various control signals to effect a state change. There are several common versions:

- Latch -- the output follows the input, but only while the clock input is "1"; lowering the clock prevents the output from changing.
- SR FF -- Set / Reset; typically created by cross-connecting two 2-input NAND gates, in which case the inputs are complemented: a "0" on the S input forces a stable "1" state, which is held until a "0" on the R input forces a "0".
- D or "delay" FF -- senses the input value at the time of a particular clock transition.
- JK FF -- the J input is an AND enable for a clocked or synchronous transition to "1"; the K input is an AND enable for a clocked transition to "0"; and often there are S and R inputs to force "1" or "0" (respectively) asynchronously.

Fourier Series

An infinite series in which the terms are constants (A, B) multiplied by sine or cosine functions of integer multiples (n) of the variable (x). One way to write this would be:

$$f(x) = A_0 + SUM\ (A_n \cos nx + B_n \sin nx)$$

Alternately, over the interval [a, a+2c]:

```
f(x) = a0 + SUM ( an cos(n PI x/c) + bn sin(n PI x/c) )
an = 1/c INTEGRAL[a,a+2c]( f(x) cos(n PI x/c) dx )
bn = 1/c INTEGRAL[a,a+2c]( f(x) sin(n PI x/c) dx )
```

Fourier Theorem

Under suitable conditions any periodic function can be represented by a Fourier series. (Various other "orthogonal functions" are now known.)

The use of sine and cosine functions is particularly interesting, since each term represents a single frequency oscillation. So to the extent that we can represent an amplitude waveform as a series of sine and cosine functions, we thus describe the frequency spectrum associated with that waveform. This frequency spectrum describes the frequencies which must be handled by a circuit to reproduce the original waveform. This illuminating computation is called a Fourier transform.

Fourier Transform

The Fourier transform relates amplitude samples at periodic discrete times to amplitude samples at periodic discrete frequencies. There are thus two representations: the amplitude vs. time waveform, and the amplitude vs. complex frequency (magnitude and phase) spectrum. Exactly the same information is present in either representation, and the transform supports converting either one into the other. This computation is efficiently performed by the FFT.

In a cryptographic context, one of the interesting parts of the Fourier transform is that it represents a thorough mixing of each input value to every output value.

Frequency

The number of repetitions or *cycles* per second. Now measured in Hertz (Hz); previously called cycles-per-second (cps).

Function

A mapping; sometimes specifically confined to numbers.

FWT

Fast Walsh Transform.

Gain

The amplitude change due to amplification. A negative gain is in fact a *loss*.

Galois Field

Finite field. First encountered by the 19-year-old student Evariste Galois, in 1830 France, a year or so before dying in a duel.

Gate

A digital logic component which is a simple logic function, possibly with a complemented output. Some common Boolean logic gates include:

- AND
- OR
- Exclusive-OR
- NAND -- AND with output complement
- NOR -- OR with output complement
- Exclusive-NOR -- Exclusive-OR with output complement
- NOT -- the complement

GF 2^n

The Galois field or finite field of 2^n polynomials of degree n-1 or less.

Typically we have mod 2 polynomials with results reduced "modulo" an irreducible "generator" polynomial *g* of degree *n*. This is analogous to creating a field from the integers modulo some prime *p*.

For example, consider GF(2^4) using the generator polynomial $x^4 + x + 1$, or 10011, which is a degree-4 irreducible. First we multiply two elements as usual:

```
  1 0 1 1
* 1 1 0 0
----------
        0
```

```
              0
      1 0 1 1
    1 0 1 1
   ---------------
    1 1 1 0 1 0 0
```

Then we "reduce" the result modulo the generator polynomial:

```
                      1 1 0
             ----------------
    1 0 0 1 1 ) 1 1 1 0 1 0 0
                1 0 0 1 1
                ---------
                  1 1 1 0 0
                  1 0 0 1 1
                  ---------
                    1 1 1 1 0
                    1 0 0 1 1
                    ---------
                        1 1 0 1
                       =========
```

So, if I did the arithmetic right, the result is the remainder, 1101. I refer to this as arithmetic "mod 2, mod p".

An irreducible is sufficient to form a finite field. However, some special irreducibles are also primitive, and these create "maximal length" sequences in LFSR's.

Goodness of Fit

In statistics, a test used to compare two distributions. For nominal or "binned" measurements, a chi-square test is common. For ordinal or ordered measurements, a Kolmogorov-Smirnov test is appropriate.

Goodness-of-fit tests can *at best* tell us whether one distribution **is** or **is not** the same as the other, and they say even *that* only with some probability. It is important to be very careful about experiment design, so that, almost always, "nothing unusual found" is the goal we seek. When we can match distributions, we are obviously able to state exactly what the experimental distribution should be and is. But there are *many* ways in which distributions can differ, and simply finding a difference is *not* evidence of a specific effect. (See null hypothesis.)

Group

In abstract algebra, a nonempty set *G* with one dyadic (two-input, one-output) operation which we choose to call "multiplication" and denote * as usual. If elements (not necessarily numbers) *a, b* are in *R*, then *ab* (or *a*b*) is also in *R*. The following properties hold:

1. **Multiplication is associative:** (ab)c = a(bc)
2. **There is a multiplicative identity:** for e in G, ea = ae = a
3. **There is a multiplicative inverse:** for a in G, there is an a^{-1} in G such that $a^{-1}a = e = aa^{-1}$

A group is basically a mapping from two elements in the group, through the group operation *m*, into the same group:

```
m:G x G -> G
```

Hamming Distance

A measure of the difference or "distance" between two binary sequences of equal length; in particular, the number of bits which differ between the sequences. This is the weight or the number of 1-bits in the exclusive-OR of the two sequences.

Hardware

The physical realization of computation. Typically, the electronic digital logic, power supply, and various electro-mechanical components such as disk drives, switches, and possibly relays which make up a computer or other digital system. As opposed to software. See system design and debug.

Hash

A classic computer operation which forms a fixed-size result from an arbitrary amount of data. Ideally, even the smallest change to the input data will change about half of the bits in the result. Often used for table look-up, so that very similar language terms or phrases will be well-distributed throughout the table. Also often used for error-detection, and, known as a message digest, authentication.

A hash of data will produce a particular hash value, which then can be included in the message before it is sent (or stored). When the data are received (or read) and the hash value computed, this should match the included hash value. So if the hash is different, something has changed, and the usual solution is to request the data be sent again. But the hash value is typically much smaller than the data, so there *must* be "many" different data sets which will produce that same value. This means that "error detection" inherently cannot detect all possible errors, and this is quite independent of any "linearity" in the hash computation.

An excellent example of a hash function is a CRC operation. CRC is a linear function without cryptographic strength, but does have a strong mathematical basis which is lacking in *ad hoc* methods. Strength is not needed when keys are processed into the state used in a random number generator, because if either the key or the state becomes known, the keyed cipher has been broken.

In contrast, a *cryptographic* hash function must be "strong" in the sense that it must be "computationally infeasible" to find two input values which produce the same hash result. In general, this means that the hash result should be 128 bits or more in size.

Sometimes a cryptographic hash function is described as being "collision free," which is a misnomer. A collision occurs when two different texts produce exactly the same hash result. Given enough texts, collisions will of course occur, precisely because any fixed-size result has only so many possible code values. The intent is that collisions be hard to find and particular hash values impossible to create at will.

Hexadecimal (Hex)

Base 16. The numerical representation in which each digit has an alphabet of sixteen symbols, generally 0 through 9, plus A through F, or "a" through "f".

Each hex value represents exactly four bits, which can be particularly convenient. Also see: binary, octal, and decimal.

Homophonic

Greek for "the same sound." The concept of having different letter sequences which are pronounced alike. In cryptography, a cipher which translates a single plaintext symbol into any one of multiple ciphertext symbols which all have the same meaning. Also see polyphonic, polygraphic and monographic.

Homophonic Substitution

A type of substitution in which an original symbol is replaced by any one of multiple unique symbols. Intended to combat the property of simple substitution in which the most-frequent symbols in the plaintext always produce the most-frequent symbols in the ciphertext.

A form of homophonic substitution is available in a large block cipher, where a homophonic selection field is enciphered along with the plaintext. Any of the possible values for that field naturally will produce a unique ciphertext. After deciphering any of those ciphertexts, the homophonic selection field could be deleted, and the exact same plaintext recovered. Note that the ability to produce a multitude of different encipherings for exactly the same data is related to the concept of a key.

IDEA

The secret key block cipher used in PGP. Designed by James Massey and Xuejia Lai in several installments, called PES, IPES and IDEA. It is round-based, with a 64-bit block size, a 128-bit key, and no internal tables.

The disturbing aspect of the IDEA design is the extensive use of *almost* linear operations, and no nonlinear tables at all. While technically *non*linear, the internal operations seem like they might well be linear *enough* to be attacked.

Ideal Secrecy

The strength delivered by even a simple cipher when each and every plaintext is equally probable and independent of every other plaintext.

There are various examples:

- The use of CBC mode in DES: By making every plaintext block equally probable, DES is greatly strengthened against codebook attack.
- The transmission of random message key values: To the extent that every value is equally probable, even a very simple cipher is sufficient to protect those values.
- The use of a keyed simple substitution *of the ciphertext* to add strength, as used in the Penknife stream cipher design.
- The use of data compression to reduce the redundancy in a message before ciphering: This of course can only *reduce* language redundancy. (Also, many compression techniques send pre-defined tables before the data and so are not suitable in this application.)

Also see: perfect secrecy. From Claude Shannon.

i.i.d.

In statistics: Independent, Identically Distributed. Generally related to the random sampling of a single distribution.

Inductive Reasoning

In the study of logic, reasoning from the observation of some particular cases to produce a general statement. While often incorrect, inductive reasoning does provide a way to go beyond known truth to new statements which may then be tested. And certain types of inductive reasoning can be assigned a correctness probability using statisticical techniques. Also see: deductive reasoning and fallacy.

Inductor

A basic electronic component which acts as a reservoir for electrical power in the form of current. An inductor thus acts to "even out" the current flowing through it, and to "emphasize" current changes across the terminals. An inductor conducts DC and opposes AC in proportion to frequency. Inductance is measured in Henrys: A voltage of 1 Volt across an inductance of 1 Henry produces a current change of 1 Ampere per Second through the inductor.

Typically a coil or multiple turns of conductor wound on a magnetic or ferrous core. Current in the conductor creates a magnetic field, thus "storing" charge. When power is removed, the magnetic field collapses to maintain the current flow; this can produce high voltages, as in automobile spark coils.

Also see capacitor and resistor.

Injective

One-to-one. A mapping f: *X* -> *Y* where no two values *x* in *X* produce the same result *f(x)* in *Y*. A one-to-one mapping is invertible for those values of *X* which produce unique results *f(x)*, but there may not be a full inverse mapping g: *Y* -> *X*.

Insulator

A material in which electron flow is difficult or impossible. Classically air or vacuum, or wood, paper, glass, ceramic, plastic, etc. As opposed to a conductor.

Integer

An element in the set consisting of *counting* numbers: 1, 2, 3, ..., their negatives: -1, -2, -3, ..., and zero.

Intermediate Block

In the context of a layered block cipher, the data values produced by one layer then used by the next.

In some realizations, an intermediate block might be wired connections between layer hardware. In the context of a general purpose computer, an intermediate block might represent the movement of data between operations, or perhaps transient storage in the original block.

Interval

In statistics, measurements in which the numerical value has meaning. Also see: nominal, and ordinal.

Into

A mapping f: *X* -> *Y* which only partially covers Y. An inverse mapping g: *Y* -> *X* may not exist if, for example, multiple elements *x* in *X* produce the same *f(x)* in *Y*.

```
+----------+          +----------+
|          |   INTO   | Y        |
|    X     |          |   +----+ |
|          |    f     |   |f(X)| |
|          |   --->   |   +----+ |
+----------+          +----------+
```

Inverse

A mapping or function *g(y)* or $f^{-1}(y)$, related to some function *f(x)* such that for each *x* in *X*:

$$g(f(x)) = x = f^{-1}(f(x)).$$

Only functions which are one-to-one can have an inverse.

Invertible

A mapping or function which has an inverse. A transformation which can be reversed.

Involution

A type of mapping which is a self-inverse.

A cipher which takes plaintext to ciphertext, and ciphertext back to plaintext, using the exact same operation.

Irreducible

A polynomial only evenly divisible by itself and 1. The polynomial analogy to integer primes. Often used to generate a *residue class field* for polynomial operations.

A polynomial form of the ever-popular "Sieve of Eratosthenes" can be used to build table of irreducibles through degree 16. That table can then be used to check any potential irreducible through degree 32. While slow, this can be a simple, clear validation of other techniques.

Also see primitive polynomial.

IV

"Initial value," "initializing value" or "initialization vector." An external value needed to start off cipher operations. Most often associated with CBC mode.

An IV often can be seen as a design-specific form of message key. Sometimes, iterative ciphering under different IV values can provide sufficient keying to perform the message key function.

Generally, an IV must be accompany the ciphertext, and so always expands the ciphertext by the size of the IV.

Jitterizer

A particular cryptographic mechanism intended to complicate the sequence produced by a linear random number generator by deleting elements from the sequence at pseudo-random.

The name is taken from the use of an oscilloscope on digital circuits, where a signal which is not "in sync" is said to "jitter." Mechanisms designed to restore synchronization are called "synchronizers," so mechanisms designed to cause jitter can legitimately be called "jitterizers."

KB

Kilobyte. 2^{10} or 1024 bytes.

Kb

Kilobit. 2^{10} or 1024 bits.

Kerckhoff's Requirements

General cryptosystem requirements formulated in 1883 (from the Handbook of Applied Cryptography):

1. **The system should be, if not theoretically unbreakable, unbreakable in practice.** (Of course there *are* no realized systems which are "theoretically unbreakable," but there is also little point in using a known breakable cipher.)
2. **Compromise of the system details should not inconvenience the correspondents.** (Nowadays we generally *assume* that the Opponent will have full details of the cipher, since, for a cipher to be widely used, it must be present at many locations and is therefore likely to be exposed. We also assume that the Opponent will have some amount of known-plaintext to work with.)
3. **The key should be rememberable without notes and easily changed.** (This is still an issue. Hashing allows us to use long language phrases, but the best approach may someday be to have both a hardware key card *and* a key phrase.)
4. **The cryptogram should be transmissible by telegraph.** (This is not very important nowadays, since even binary ciphertext can be converted into ASCII for transmission if necessary.)
5. **The encryption apparatus should be portable and operable by a single person.** (Software encryption approaches this ideal.)
6. **The system should be easy, requiring neither the knowledge of a long list of rules nor mental strain.** (Software encryption has the *potential* to approach this, but often fails to do so. We might think of the need to certify public keys, which is still often left up to the user, and thus often does not occur.)

Key

The general concept of protecting things with a "lock," thus making those things available only if one has the correct "key." In a cipher, the ability to select a particular transformation between a plaintext message and a corresponding ciphertext. By using a particular key, we can create any one of many different ciphertexts for the exact same message. And if we know the correct key, we can transform the ciphertext back into the original message. By supporting a vast number of different key possibilities (a large keyspace), we hope to make it impossible for someone to decipher the message by trying every key in a brute force attack.

In cryptography we have various kinds of keys, including a User Key (the key which a user actually remembers), which may be the same as an Alias Key (the key for an alias file which relates correspondent names with their individual keys). We may also have an Individual Key (the key actually used for a particular correspondent); a Message Key (normally a random value which differs for each and every message); a Running Key (the confusion sequence in a stream cipher, normally produced by a random number generator); and perhaps other forms of key as well.

In general, the value of a cryptographic key is used to initialize the state of a cryptographic mechanism.

Ideally, a key will be a equiprobable selection among a huge number of possibilities. This is the fundamental strength of cryptography, the "needle in a haystack" of false possibilities. But if a key is in some way *not* a random selection, but is instead *biased*, the most-likely keys can be examined first, thus reducing the complexity of the search and the effective keyspace.

In most cases, a key will exhibit diffusion across the message; that is, changing even one bit of a key should change every bit in the message with probability 0.5. A key with lesser diffusion may succumb to some sort of divide and conquer attack.

Key Distribution Problem

The problem of distributing keys to both ends of a communication path, especially in the case of secret key ciphers, since secret keys must be transported and held in absolute secrecy. Also the problem of distributing vast numbers of keys, if each user is given a separate key.

Although this problem is supposedly "solved" by the advent of the public key cipher, in fact, the necessary public key validation is almost as difficult as the original problem. Although public keys can be *exposed*, they must represent who they claim to represent, or a "spoofer" or man-in-the-middle can operate undetected.

Nor does it make sense to give each individual a separate secret key, when a related group of people would have access to the same files anyway. Typically, a particular group has the same secret key, which will of course be changed when any member leaves. Typically, each individual would have a secret key for each group with whom he or she associates.

Keyspace

The number of distinct key-selected transformations supported by a particular cipher. Normally described in terms of bits, as in the number of bits needed to count every distinct key. This is also the amount of state required to support a state value for each key. The keyspace in bits is the $\log_2$ (the base-2 logarithm) of the number of different keys, provided that all keys are equally probable.

Cryptography is based on the idea that if we have a huge number of keys, and select one at random, The Opponents generally must search about half of the possible keys to find the correct one; this is a brute force attack.

Although brute force is not the only possible attack, it is the one attack which will always exist. Therefore, the ability to resist a brute force attack is normally the "design strength" of a cipher. All other attacks should be made even more expensive. To make a brute force attack expensive, a cipher simply needs a keyspace large enough to resist such an attack. Of course, a brute force attack may use new computational technologies such as DNA or "molecular computation." Currently, 120 bits is large enough to prevent even unimaginably large uses of such new technology.

It is probably just as easy to build efficient ciphers which use huge keys as it is to build ciphers which use small keys, and the cost of storing huge keys is probably trivial. Thus, large keys may be useful when this leads to a better cipher design, perhaps with less key processing. Such keys, however, cannot be considered better at resisting a brute force attack than a 120-bit key, since 120 bits is already sufficient.

Keyed Substitution

Two substitution tables of the same size with the same values can differ only in the ordering or permutation of the values in the tables. A huge keying potential exists: The typical "n-bit-wide" substitution table has 2^n elements, and $(2^n)!$ ("two to the nth factorial") different permutations or key possibilities. A single 8-bit substitution table has a keyspace of 1648 bits.

A substitution table is keyed by creating a *particular* ordering from each different key. This can be accomplished by shuffling the table under the control of a random number generator which is initialized from the key.

Known Plaintext Attack

A type of attack in which the cryptanalyst has some quantity of related plaintext and ciphertext. This allows the ciphering transformation to be examined directly.

A known plaintext attack is especially dangerous to the usual stream cipher which has an additive combiner, because the known plaintext can be "subtracted" from the ciphertext, thus completely exposing the confusion sequence. This is the sequence produced by the cryptographic random number generator, and can be used to attack that generator. This sort of attack can generally be prevented by using a Dynamic Substitution Combiner instead of the usual additive combiner.

It is surprisingly reasonable that The Opponent might well have some known plaintext (and related ciphertext): This might be the return address on a letter, a known report, or even some suspected words. Sometimes the cryptosystem will carry unauthorized messages like birthday greetings which are then exposed, due to their apparently innocuous content.

Kolmogorov-Smirnov

In statistics, a goodness of fit test used to compare two distributions of ordinal data, where measurements may be re-arranged and placed in order. Also see chi-square.

- n independent samples are collected and arranged in numerical order in array X as $x[0]..x[n-1]$.
- $S(x[j])$ is the fraction of the n observations which are less than or equal to $x[j]$; in the ordered array this is just $((j+1)/n)$.
- $F(x)$ is the reference cumulative distribution, the probability that a random value will be less than or equal to x. Here we want $F(x[j])$, the fraction of the distribution to the left of $x[j]$ which is a value from the array.

The "one-sided" statistics are:

```
K+ = SQRT(N) * MAX( S(x[j]) - F(x[j]) )
   = SQRT(N) * MAX( ((j+1)/n) - F(x[j]) )

K- = SQRT(N) * MAX( F(x[j]) - S(x[j]) )
   = SQRT(N) * MAX( F(x[j]) - (j/n) )
```

And the "two-sided" KS statistic is:

```
K = SQRT(N) * MAX( ABS( S(x[j]) - F(x[j]) ) )
```

$= MAX(K^+, K^-)$

It appears that the "one-sided" KS distribution is far easier to compute precisely, and may be preferred on that basis.

See the Kolmogorov-Smirnov section of the Ciphers By Ritter / JavaScript computation pages.

Latency

A form of delay. Typically a hardware term, latency often refers to the time need to perform an operation. In the past, operation delay has largely been dominated by the time taken for gate switching transistors to turn on and off. Currently, operation delay is more often dominated by the time it takes to transport the electrical signals to and from gates on long, thin conductors.

The effect of latency on throughput can often be reduced by *pipelining* or partitioning the main operation into many small sub-operations, and running each of those *in parallel*, or at the same time. As each operation finishes, that result is latched and saved temporarily, pending the availability of the next sub-operation hardware. The result is throughput limited only by the longest sub-operation instead of the overall operation.

Latin Square

A Latin square of order *n* is an *n* by *n* array containing symbols from some alphabet of size *n*, arranged such that each symbol appears exactly once in each row and exactly once in each column. Also see Latin square combiner and orthogonal Latin squares.

```
2  0  1  3
1  3  0  2
0  2  3  1
3  1  2  0
```

Also see: Latin Squares: A Literature Survey, in the Literature Surveys and Reviews section of the Ciphers By Ritter page.

Latin Square Combiner

A cryptographic combining mechanism in which one input selects a column and the other input selects a row in an existing Latin square; the value of the selected element is the combiner result.

A Latin square combiner is inherently balanced, because for any particular value of one input, the other input can produce any possible output value. A Latin square can be treated as an array of substitution tables, each of which are invertible, and so can be reversed for use in a suitable extractor. As usual with cryptographic combiners, if we know the output and a specific one of the inputs, we can extract the value of the other input.

For example, a tiny Latin square combiner might combine two 2-bit values each having the range zero to three (0..3). That Latin square would contain four different symbols (here 0, 1, 2, and 3), and thus be a square of order 4:

```
2  0  1  3
1  3  0  2
0  2  3  1
3  1  2  0
```

With this square we can combine the values 0 and 2 by selecting the top row (row 0) and the third column (column 2) and returning the value 1.

When extracting, we will know a specific one (but only one) of the two input values, and the result value. Suppose we know that row 0 was selected during combining, and that the output was 1: We can check for the value 1 in each column at row 0 and find column 2, but this involves searching through all columns. We can avoid this overhead by creating the row-inverse of the original Latin square (the inverse of each row), in the well-known way we would create the inverse of any invertible substitution. For example, in row 0 of the original square, selection 0 is the value 2, so, in the row-inverse square, selection 2 should be the value 0, and so on:

```
1  2  0  3
2  0  3  1
0  3  1  2
3  1  2  0
```

Then, knowing we are in row 0, the value 1 is used to select the second column, returning the unknown original value of 2.

A practical Latin square combiner might combine two bytes, and thus be a square of order 256, with 65,536 byte entries. In such a square, each 256-element column and each 256-element row would contain each of the values from 0 through 255 exactly once.

Layer

In the context of block cipher design, a layer is particular transformation or set of operations applied across the block. In general, a layer is applied once, and different layers have different transformations. As opposed to rounds, where a single transformation is repeated in each round.

Layers can be confusion layers (which simply change the block value), diffusion layers (which propagate changes across the block in at least one direction) or both. In some cases it is useful to do multiple operations as a single layer to avoid the need for internal temporary storage blocks.

LFSR

Linear Feedback Shift Register.

Linear

Like a line; having an equation of the form **ax + b** .

There are various ways a relationship can be linear. One way is to consider *a*, *x*, and *b* as integers. Another is for them to be polynomial elements of $GF(2^n)$. Yet another is to consider **a** to be an *n* by *n* matrix, with **x** and **b** as *n*-element vectors. There are probably various other ways as well.

Linearity also depends upon our point of view: For example, integer addition *is* linear in the integers, but when expressed as mod 2 operations, the exact same computation producing the exact same results is *not* considered linear.

In cryptography the issue may not be as much one of strict mathematical linearity as it is the "distance" between a function and some linear approximation (see Boolean function nonlinearity). True linear functions are used because they are easy and fast, but they are also exceedingly weak. Of course XOR is linear and trivial, yet is used all the time in arguably strong ciphers. But a design using linear components must have other nonlinear components to provide strength.

Linear Complexity

The length of the shortest Linear Feedback Shift Register which can produce a given sequence.

Also see: Linear Complexity: A Literature Survey, in the Literature Surveys and Reviews section of the Ciphers By Ritter page.

Linear Feedback Shift Register

An efficient structure for producing sequences, often used in random number generator applications.

In an *n*-element shift register (SR), if the last element is connected to the first element, a set of *n* values can circulate around the SR in *n* steps. But if the values in two of the elements are combined by exclusive-OR and that result connected to the first element, it is possible to get an almost-perfect maximal length sequence of 2^n-1 steps. (The all-zeros state will produce another all-zeros state, and so the system will "lock up" in a degenerate cycle.) Because there are only 2^n different states of *n* binary values, every state value but one must occur exactly once, which is a statistically-satisfying result. Moreover, the values so produced are a perfect permutation of the "counting" numbers (1..2^n-1).

```
A Linear Feedback Shift Register

    +----+    +----+    +----+    +----+    +----+  "a0"
+-<-| a5 |<---| a4 |<-*-| a3 |<---| a2 |<---| a1 |<--+
|   +----+    +----+  | +----+    +----+    +----+   |
|                     v                               |
+------------------> (+) -----------------------------+
```

```
    1         0         1         0         0         1
```

In the figure we have a LFSR of degree 5, consisting of 5 storage elements a[5]..a[1] and the feedback computation a[0]=a[5]+a[3]. The stored values may be bits and the operation (+) addition mod 2. A clock edge will simultaneously shift all elements left, and load element a[1] with the feedback result as it was before the clock changed the register. Each SR element is just a time-delayed replica of the element before it, and here the element subscript conveniently corresponds to the delay. We can describe this logically:

```
a[1][t+1] = a[5][t] + a[3][t];
a[2][t+1] = a[1][t];
a[3][t+1] = a[2][t];
a[4][t+1] = a[3][t];
a[5][t+1] = a[4][t];
```

Normally the time distinction is ignored, and we can write more generally, for some feedback polynomial C and state polynomial A of degree *n*:

```
         n
a[0] = SUM  c[i]*a[i]
       i=1
```

The feedback polynomial shown here is 101001, a degree-5 poly running from c[5]..c[0] which is also irreducible. Since we have degree 5 which is a Mersenne prime, C is also primitive. So C produces a maximal length sequence of exactly 31 steps, provided only that A is not initialized as zero. Whenever C is irreducible, the reversed polynomial (here 100101) is also irreducible, and will also produce a maximal length sequence.

LFSR's are often used to generate the confusion sequence for stream ciphers, but this is very dangerous: LFSR's are inherently linear and thus weak. Knowledge of the feedback polynomial and only *n* element values (from known plaintext) is sufficient to run the sequence backward or forward. And knowledge of only 2*n* elements is sufficient to develop an unknown feedback polynomial. This means that LFSR's should not be used as stream ciphers without in some way isolating the sequence from analysis. Also see jitterizer and additive RNG.

Linear Logic Function

A Boolean switching or logic function which can be realized using only XOR and AND types of functions, which correspond to addition mod 2 and multiplication mod 2, respectively.

Logic

A branch of philosophy related to distinguishing between correct and incorrect reasoning. Even an invalid argument can sometimes produce a correct conclusion. But a *valid* argument must *always* produce a correct conclusion.

Also devices which realize symbolic logic, such as Boolean logic, a logic of TRUE or FALSE values. Also see: subjective, objective, contextual, absolute, inductive reasoning, deductive reasoning, and fallacy.

Logic Function

Fundamental digial logic operations. The fundamental two-input (*dyadic*) one-output Boolean functions are AND and OR. The fundamental one-input (*monadic*) one-output operation is NOT. These can be used in various ways to build exclusive-OR (XOR), which is also widely used as a fundamental function. Here we show the truth tables for the fundamental functions:

```
INPUT    NOT
   0      1
   1      0

INPUT    AND   OR    XOR
 0 0      0     0     0
 0 1      0     1     1
 1 0      0     1     1
 1 1      1     1     0
```

These Boolean values can be stored as a bit, and can be associated with 0 or 1, FALSE or TRUE, NO or YES, etc.

LSB

Least-Significant Bit. Typically the rightmost bit.

M-Sequence

A maximal length shift register sequence.

Machine Language

Also "machine code." A computer program in the form of the numeric values or "operation codes" ("opcodes") which the computer can directly execute as instructions, commands, or "orders." Thus, the very public code associated with the instructions available in a particular computer. Also the programming of a computer at the bit or hexadecimal level, below even assembly language. Also see source code and object code.

Magnetic Field

The fundamental physical force resulting from moving charges. Also see: electromagnetic field.

Man-in-the-Middle Attack

The original model used to analyze cryptosystems assumed that an Opponent could *listen* to the ciphertext traffic, and perhaps even *interfere* with it, but not that messages could be intercepted and completely hidden. Unfortunately, this is in fact the situation in a store-and-forward computer network like the Internet. Routing is not secure on the Internet, and it is at least conceivable that messages between two people could be routed through connections on the other side of the world. This might be exploited to make such messages flow through a particular computer for special processing.

The Man-in-the-Middle (MITM) Attack is mainly applicable to public key systems, and focuses on the idea that many people will send their public keys on the network. The bad part of this is a lack of key authentication, because the Man-in-the-Middle can send a key just as easily, and pretend to be the other end. Then, if one *uses* that key, one has secure communication with The Opponent, instead of the far end. The MITM can receive a message, decipher it, read it, re-encipher it in the correct public key, and send it along. In this way, neither end need know anything is wrong, yet The Opponent is reading the mail.

Perhaps the worst part of this is that a successful MITM attack does not involve *any* attack on the actual ciphering. And this means that all proofs or confidence in the security of particular ciphering mechanisms is totally irrelevant to the security of a system which supports MITM attacks.

The way to avoid MITM attacks is to *certify* public keys, but this is inconvenient and time-consuming. Unless the cipher *requires* keys to be certified, this is rarely done. The worst part of this is that a successful MITM attack consumes few resources, need not "break" the cipher itself, and may provide just the kind of white-collar desktop intelligence a bureaucracy would love.

It is interesting to note that, regardless of how inconvenient it may be to share keys for a secret-key cipher, this is an inherent authentication which prevents MITM attacks.

Mapping

Given sets *X* and *Y*, and operation *f*

```
f: X -> Y  ,
```

the *mapping* or *function* or *transformation f* takes any value in the domain *X* into some value in the range, which is contained in *Y*. For each element *x* in *X*, a mapping associates a single element *y* in *Y*. Element *f(x)* in *Y* is the *image* of element *x* in *X*.

- If *f(X)* covers all elements in *Y*, *f* is a mapping of *X* **onto** *Y*, and is **surjective**.
- If *f(X)* only partially covers *Y*, *f* is a mapping of *X* **into** *Y*.

If no two values of *x* in *X* produce the same result *f(x)*, *f* is **one-to-one** or **injective**.

- If *f* is both injective and surjective, it is **one-to-one** **and** **onto** or **bijective**.

- If f is bijective, there exists an **inverse** f^{-1} such that:

  $$f^{-1}(f(x)) = x.$$

- If f is identical with f^{-1}, f is an **involution**.
- A permutation of X is a bijection from X to X.

Markov Process

In statistics, a stochastic (random) process (function) in which all possible outcomes are defined by the current state, independent of all previous states. Also see: stationary process.

Mathematical Cryptography

Cryptography based on mathematical operations, such as taking extremely large values to extremely large powers, modulo the product of two primes. Normally heavily involved with number theory. As opposed to mechanistic cryptography.

There are some problems with a strictly mathematical approach to cryptography:

1. Mathematical symbology has evolved for concise expression. It is thus not "isomorphic" to the complexity of the implementation, and so is not a good vehicle for the design-time trade-off of computation versus strength.
2. Most mathematical operations are useful or "beautiful" relationships specifically intended to support understanding in either direction, as opposed to relationships which might be particularly difficult to reverse or infer. So when using the traditional operations for cryptography, we must first defeat the very properties which made these operations so valuable in their normal use.
3. Mathematics has evolved to produce, describe and expose *structure*, as in useful or "beautiful" large-scale relationships and groupings. But, in a sense, relationships and groupings are the exact opposite of the fine-grained completely random mappings that cryptography would like to see. Such mappings are awkward to express mathematically, and contain little of the structure which mathematics is intended to describe.
4. There may be an ingrained tendency in math practitioners, based on long practice, to *construct* math-like relationships, and such relationships are not desirable in this application. So when using math to construct cryptography, we may first have to defeat our own training and tendencies to group, understand and simplify.

On the other hand, mathematics is *irreplaceable* in providing the tools to pick out and describe *structure* in apparently strong cipher designs. Mathematics can identify specific strength problems, and evaluate potential fixes. But there appears to be no real hope of evaluating strength with respect to *every possible* attack, even using mathematics.

Although mathematical cryptography has held out the promise of providing *provable* security, in over 50 years of work, **no** practical cipher has been generally accepted as having *proven* strength. See, for example: one time pad.

MB

Megabyte. 2^{20} or 1,048,576 bytes.

Mb

Megabit. 2^{20} or 1,048,576 bits.

Maximal Length

A linear feedback shift register (LFSR) sequence of 2^n-1 steps (assuming a bit-wide shift register of n bits. This means that every binary value the register can hold, except zero, will occur on some step, and then not occur again until all other values have been produced. A maximal-length LFSR can be considered a binary counter in which the count values have been shuffled or enciphered. And while the sequence from a normal binary counter is perfectly balanced, the sequence from a maximal-length LFSR is *almost* perfectly balanced. Also see M-sequence.

Mechanism

The logical concept of a machine, which may be realized either as a physical machine, or as a sequence of logical commands executed by a physical machine.

A mechanism can be seen as a process or an implementation for performing that process (such as electronic hardware, computer software, hybrids, or the like).

Mechanistic Cryptography

Cryptography based on mechanisms, or machines. As opposed to mathematical cryptography.

Although perhaps looked down upon by those of the mathematical cryptography persuasion, mechanistic cryptography certainly does use mathematics to design and predict performance. But rather than being restricted to arithmetic operations, mechanistic cryptography tends to use a wide variety of mechanically-simple components which may not have concise mathematical descriptions. Rather than simply implementing a system of math expressions, complexity is constructed from the various efficient components available to digital computation.

Mersenne Prime

A prime p for which 2^p - 1 is also prime. For example, 5 is a Mersenne prime because 2^5 - 1 =
31, and 31 is prime. For mod 2 polynomials of Mersenne prime degree, *every* irreducible is also primitive.

```
Mersenne Primes:
    2         107          9689         216091
    3         127          9941         756839
    5         521         11213         859433
    7         607         19937        1257787
   13        1279         21701        1398269
   17        2203         23209
   19        2281         44497
   31        3217         86243
   61        4253        110503
   89        4423        132049
```

Message Digest

A small value which represents an entire message for purposes of authentication; a hash.

Message Key

A key transported with the message and used for deciphering the message. (The idea of a "session key" is very similar, but lasts across multiple messages.)

Normally, the message key is a large random value which becomes the key for ciphering the data in a single message. Normally, the message key itself is enciphered under the User Key or other key for that link. The receiving end first deciphers the message key, then uses that value as the key for deciphering the message data. Alternately, the random value itself may be sent unenciphered, but is then enciphered or hashed (under a keyed cryptographic hash) to produce a value used as the data ciphering key.

The message key assures that the actual data is ciphered under a key which is an arbitrary selection from a huge number of possible keys; it therefore prevents weakness due to user key selection. A message key is used exactly once, no matter how many times the same message is enciphered, so at most, a successful attack on a message key exposes just one message. The internal construction of a random message key cannot be controlled by a user, and thus prevents all attacks based on repeated ciphering under a single key. To the extent that the message key value really is random and is never exposed on either end, the message key is much more easily protected than ordinary text (see ideal secrecy). In a sense, a message key is the higher-level concept of an IV, which is necessarily distinct for each particular design.

MITM

Man In The Middle.

Mixing

The act of transforming multiple input values into one or more output values, such that changing any input value will change the output value. There is no implication that the result must be balanced, but effective mixing may need to be, in some sense, complete. Also see Mixing Cipher, combiner, Latin square combiner, and Balanced Block Mixing.

Mixing Cipher

A block cipher based on Balanced Block Mixing of small elements in FFT-like or FWT-like mixing patterns.

Below, we have a toy 32-bit-block Mixing Cipher. Plaintext at the top is transformed into ciphertext at the bottom. Each "S" is an 8-bit substitution table, and each table (and now each mixing operation also) is individually keyed.

Horizontal lines connect elements which are to be mixed together: Each *---* represents a single Balanced Block Mixing or BBM. Each BBM takes two elements, mixes them, and returns two mixed values. The mixed results then replace the original values in the selected positions just like the "butterfly" operations used in some FFT's.

```
A 32-Bit Mixing Cipher

 |   |   |   |    <- Input Block (Plaintext)
 S   S   S   S    <- Fencing
 |   |   |   |
 *---*   *---*    <- 2 BBM Mixings
 |   |   |   |
 *-------*   |    <- 1 BBM Mixing
 |   *-------*    <- 1 BBM Mixing
 |   |   |   |
 S   S   S   S    <- Fencing
 |   |   |   |
 *-------*   |
 |   *-------*
 |   |   |   |
 *---*   *---*
 |   |   |   |
 S   S   S   S    <- Fencing
 |   |   |   |    <- Output Block (Ciphertext)
```

By mixing each element with another, and then each pair with another pair and so on, every element is eventually mixed with every other element. Each BBM mixing is dyadic, so each "sub-level" is a mixing of twice as many elements as the sublevel before it. A block of *n* elements is thus fully mixed in $\log_2 n$ sublevels, and each result element is equally influenced equally by each and every input element.

The pattern of these mixings is exactly like some implementations of the FFT, and thus the term "FFT-style." Also see the articles in the Mixing Ciphers section on the Ciphers By Ritter pages.

Mod 2

The field formed from the set of integers {0,1} with operations + and * producing the remainder after dividing by modulus 2. Thus:

```
0 + 0 = 0
0 + 1 = 1
1 + 0 = 1
1 + 1 = 0

1 + 1 + 1 = 1

0 * 0 = 0
0 * 1 = 0
1 * 0 = 0
1 * 1 = 1
```

Subtraction mod 2 is the same as addition mod 2. The operations + and * can also be considered the logic functions XOR and AND respectively.

Mod 2 Polynomial

A polynomial in which the coefficients are taken mod 2. The four arithmetic operations addition, subtraction, multiplication and division are supported. As usual, mod 2 subtraction is the same as mod 2 addition. Each column of coefficients is added separately, without "carrys" to an adjacent column:

```
Addition and Subtraction:

      1 0 1 1
    + 0 1 0 1
    + 1 1 0 0
     ---------
      0 0 1 0


Multiplication:

        1 0 1 1
      * 1 1 0 0
      ----------
              0
            0
    1 0 1 1
  1 0 1 1
 ---------------
  1 1 1 0 1 0 0
```

Polynomial multiplication is *not* the same as repeated polynomial addition. But there is a fast approach to squaring mod 2 polynomials:

```
              a  b  c  d
             ------------
             ad bd cd dd
          ac bc cc dc
       ab bb cb db
    aa ba ca da
   ----------------------
     a  0  b  0  c  0  d
```

To square a mod 2 polynomial, all we have to do is "insert" a zero between every column. Note that aa = a for a = 0 or a = 1, and ab = ba, so either 0 + 0 = 0 or 1 + 1 = 0.

```
Division:
                  1 0 1 1
            ----------------
    1 1 0 0 ) 1 1 1 0 1 0 0
              1 1 0 0
             ---------
                 1 0 1 0
                 1 1 0 0
                ---------
                   1 1 0 0
```

```
 1 1 0 0
---------
       0
```

The decision about whether the divisor "goes into" the dividend is based exclusively on the most-significant (leftmost) digit. This makes polynomial division far easier than integer division.

Mod 2 polynomials behave much like integers in that one polynomial may or may not divide another without remainder. This means that we can expect to find analogies to integer "primes," which we call *irreducible* polynomials.

Mod 2 polynomials do not constitute a field; clearly, the size of a multiplication is unbounded. However, a finite field of polynomials can be created by choosing an irreducible modulus polynomial, thus producing a Galois field GF 2^n.

Mode

One possibility is: block cipher operating mode.

Modulo

Casually, the remainder after an integer division by a modulus; see congruence. When the modulus is prime, this may generate a useful field.

Monadic

Relating to *monad*, which is Greek for single or one. In particular, a function with a single input or argument, also called unary. Also see: dyadic.

Monoalphabetic Substitution

Substitution using a single alphabet. Also called simple substitution. As opposed to Polyalphabetic Substitution.

Monographic

Greek for "single letter." A cipher which translates one plaintext symbol at a time into ciphertext. As opposed to polygraphic; also see homophonic and polyphonic.

Multiple Encryption

Enciphering or encrypting a message more than once. This usually has the strength advantage of producing a very random-like ciphertext from the first pass, which is of course the "plaintext" for the next pass.

Multiple encryption using different keys can be a way to increase strength. And multiple encryption using different ciphers can reduce the probability of using a single cipher which has been broken in secret. In both cases, the cost is additional ciphering operations.

Unfortunately, multiple encryption using just *two* (2) ciphers may not be much advantage: If we assume The Opponents know which ciphers are used, they can manipulate *both* the plaintext *and* the ciphertext to search for a match (a "meet-in-the-middle" attack strategy). One way to avoid this is to use *three* (3) cipherings, as in Triple DES.

Multiple encryption also can be *dangerous*, if a single cipher is used with the same key each time. Some ciphers are involutions which both encipher and decipher with the same process; these ciphers will **de**cipher a message if it is **en**ciphered a second time under the same key. This is typical of classic additive synchronous stream ciphers, as it avoids the need to have separate encipher and decipher operations. But it also can occur with block ciphers operated in stream-cipher-like modes such as OFB, for exactly the same reason.

Nomenclator

Originally, a list of transformations from *names* to symbols or numbers for diplomatic communications. Later, typically a list of transformations from names, polygraphic syllables, and monographic letters, to numbers. Usually the monographic transformations had multiple or homophonic alternatives for frequently-used letters. Generally smaller than a codebook, due to the use of the syllables instead of a comprehensive list of phrases. A sort of early manual cipher with some characteristics of a code, that operated like a codebook.

Nominal

In statistics, measurements which are in categories or "bins." Also see: ordinal, and interval.

Nonlinearity

The extent to which a function is not linear. See Boolean function nonlinearity.

NOT

A Boolean logic function which is the "complement" or the mod 2 addition of 1.

Null Hypothesis

In statistics, the particular statement or hypothesis H_0 which is accepted unless a statistic testing that hypothesis produces evidence to the contrary. Normally, the null hypothesis is accepted when the associated statistical test indicates "nothing unusual found."

The logically contrary alternative hypothesis H_1 is sometimes formulated with the specific *hope* that something unusual *will* be found, but this can be very tricky to get right. Many statistical tests (such as goodness-of-fit tests) can only indicate whether something matches what we expect, or does not. But any number of things can cause a mismatch, including a fundamentally flawed experiment. A simple mismatch does not normally imply the presence of a particular quality.

Even in the best possible situation, random sampling will produce a range or distribution of test statistic values. Often, even the worst possible statistic value can be produced by an unlucky sampling of the best possible data. It is thus important to know what distribution to expect because of the sampling alone, so if we find a *different* distribution, that will be evidence supporting the alternative hypothesis H_1.

If we collect enough statistic values, we should see them occur in the ideal distribution for that particular statistic. So if we call the upper 5 percent of the distribution "failure" (this is the significance level) we not only *expect* but in fact *require* such "failure" to occur about 1 time in 20. If it does not, we will in fact have detected something unusual, something which might even indicate problems in the experimental design.

If we have only a small number of samples, and do not run repeated trials, a relatively few chance events can produce an improbable statistic value, which might cause us to reject a valid null hypothesis, and so commit a type I error.

On the other hand, if there *is* a systematic deviation in the underlying distribution, only a very specific type of random sampling could mask that problem. With few samples and trials, though, the chance random masking of a systematic problem is still *possible*, and could lead to a type II error.

Object Code

Typically, machine language instructions represented in a form which can be "linked" with other routines. Also see source code.

Objective

In the study of logic, reality observed without interpretation. As opposed to subjective or interpreted reality. Alternately, a goal.

Octal

Base 8: The numerical representation in which each digit has an alphabet of eight symbols, generally 0 through 7.

Somewhat easier to learn than hexadecimal, since no new numeric symbols are needed, but octal can only represent three bits at a time. This generally means that the leading digit will not take all values, and that means that the representation of the top part of two concatenated values will differ from its representation alone, which can be confusing. Also see: binary and decimal.

Octave

A frequency ratio of 2:1. From an 8-step musical scale.

OFB

OFB or Output FeedBack is an operating mode for a block cipher.

OFB is closely related to CFB, and is intended to provide some of the characteristics of a stream cipher from a block cipher. OFB is a way of using a block cipher to form a random number generator. The resulting pseudorandom confusion sequence can be combined with data as in the usual stream cipher.

OFB assumes a shift register of the block cipher block size. An IV or initial value first fills the register, and then is ciphered. Part of the result, often just a single byte, is used to cipher data, and *also* is shifted into the register. The resulting new register value is ciphered, producing another confusion value for use in stream ciphering.

One disadvantage of this, of course, is the need for a full block-wide ciphering operation, typically for each data byte ciphered. The advantage is the ability to cipher individual characters, instead of requiring accumulation into a block before processing.

One Time Pad

The term "one time pad" (OTP) is rather casually used for two fundamentally different types of cipher:

1. **The Theoretical One Time Pad:** a theoretical random source produces values which are combined with data to produce ciphertext. In a theoretical discussion of this concept, we can simply **assume** *perfect* randomness in the source, and this **assumption** supports a mathematical proof that the cipher is unbreakable. But the theoretical result applies to reality **only if we can prove the assumption is valid** in reality. Unfortunately, we cannot do this, because *provably* perfect randomness apparently cannot be attained in practice. So the theoretical OTP does not really exist, except as a goal.
2. **The Realized One Time Pad:** a really random source produces values which are combined with data to produce ciphertext. But because we can neither *assume* nor *prove* perfect, theoretical-class randomness in any real generator, this cipher does not have the mathematical proof of the theoretical system. Thus, a realized one time pad is **NOT *proven*** unbreakable, although it may in fact *be* unbreakable in practice. In this sense, it is much like other realized ciphers.

A realized one time pad (OTP) is essentially a stream cipher with a really random confusion sequence used exactly once. The confusion sequence is the key, and it is as long as the data. Since this amount of keying material can be awkward to transfer and keep, we often see "pseudo" one-time pad designs which attempt to correct this deficiency. Normally, the point is to achieve the theoretical advantages of a one-time pad without the costs; the problem with this is that the one-time pad theory of strength no longer applies. These variations are best seen as classic stream cipher designs.

In a realized one time pad, the confusion sequence must be *unpredictable* (not generated from a small key value) and must be transported to the far end and held at both locations in absolute secrecy like any other secret key. But where a normal secret key might range perhaps from 16 bytes to 160 bytes, there must be as much OTP sequence as there will be data (which might well be megabytes). And a normal secret key could itself be sent under a key (as in a message key or under a public key). But an OTP sequence *cannot* be sent under a key, since this would make the OTP as weak as the key, in which case we might as well use a normal cipher. All this implies very significant inconveniences, costs, and risks, well beyond what one would at first expect, so even the realized one time pad is generally considered **impractical**, except in very special situations.

In a realized one time pad, the confusion sequence itself must be random for, if not, it will be somewhat predictable. And, although we have a great many statistical randomness tests, there is no test which can *certify* a sequence as either random or unpredictable. This means that a sequence which we assume to be random may **not** be the unpredictable sequence we need, and we can never know for sure. (This might be considered an argument for using a combiner with strength, such as a Latin square or Dynamic Substitution.) In practice, the much touted "mathematically proven unbreakability" of the one time pad depends upon an assumption of randomness and unpredictability which we can neither test nor prove.

The one time pad sometimes seems to have yet another level of strength above the usual stream cipher, the ever-increasing amount of "unpredictability" or entropy in the confusion sequence, leading to an indefinite unicity distance. In contrast, the typical stream cipher will produce a long sequence from a relatively small amount of initial state, and it can be argued that the entropy of an RNG is just the number of bits in its initial state. In theory, this might mean that the initial state or key used in the stream cipher could be identified after somewhat more than that same amount of data had been enciphered. But it is also perfectly possible for an unsuspected problem to occur in a really-random generator, and then the more sequence generated, the more apparent and useful that problem might be to an Opponent.

Nor does even a theoretical one time pad imply unconditional security: Consider *A* sending the same message to *B* and *C*, using, of course, two *different* pads. Now, suppose the Opponents can acquire plaintext from *B* and intercept the ciphertext to *C*. If the system is using the usual additive combiner, the Opponents can *reconstruct the pad* between *A* and *C*. Now they can send *C* any message they want, and encipher it under the correct pad. And *C* will never question such a message, since *everyone knows* that a one time pad provides "absolute" security as long as the pad is kept secure. Note that both *A* and *C* have done this, and they are the only ones who had that pad.

Various companies offer one time pad programs, and sometimes also the keying or "pad" material.

One-To-One

Injective. A mapping f: *X* -> *Y* where no two values *x* in *X* produce the same result *f(x)* in *Y*. A one-to-one mapping is invertible for those values of *X* which produce unique results *f(x)*, but there may not be a full inverse mapping g: *Y* -> *X*.

One Way Diffusion

In the context of a block cipher, a one way diffusion layer will carry any changes in the data block in a direction from one side of the block to the other, but not in the opposite direction. This is the usual situation for fast, effective diffusion layer realizations.

Onto

Surjective. A mapping f: *X* -> *Y* where *f(x)* covers all elements in *Y*. Not necessarily invertible, since multiple elements *x* in *X* could produce the same *f(x)* in *Y*.

```
+----------+          +----------+
|          |   ONTO   |          |
|    X     |          | Y = f(X) |
|          |    f     |          |
|          |   --->   |          |
+----------+          +----------+
```

Opcode

Operation code: a value which selects one operation from among a set of possible operations. This is an encoding of functions as values. These values may be interpreted by a computer to perform the selected operations in their given sequence and produce a desired result. Also see: software and hardware.

Operating Mode

With respect to block ciphers, a way to handle messages which are larger than the defined block size. Usually this means one of the four block cipher "applications" defined for use with DES:

- ECB or Electronic CodeBook;
- CBC or Cipher Block Chaining;
- CFB or Ciphertext FeedBack; and
- OFB or Output FeedBack.

It can be argued that block cipher operating modes are stream "meta-ciphers" in which the streamed transformation is of full block cipher width, instead of the usual stream cipher bit- or byte-width transformations.

Opponent

A term used by some cryptographers to refer to the opposing cryptanalyst or opposing team. Sometimes used in preference to "the enemy."

OR

A Boolean logic function which is also nonlinear under mod 2 addition.

Order

In mathematics, typically the number of elements in a structure, or the number of steps required to traverse a cyclic structure.

Ordinal

In statistics, measurements which are ordered from smallest to largest. Also see: nominal, and interval.

Orthogonal

At right angles; on an independent dimension. Two structures which each express an independent dimension.

Orthogonal Latin Squares

Two Latin squares of order *n*, which, when superimposed, form each of the n^2 possible ordered pairs of *n* symbols exactly once. At most, *n*-1 Latin squares may be mutually orthogonal.

```
3 1 2 0   0 3 2 1       30  13  22  01
0 2 1 3   2 1 0 3   =   02  21  10  33
1 3 0 2   1 2 3 0       11  32  03  20
2 0 3 1   3 0 1 2       23  00  31  12
```

Also see Balanced Block Mixing.

OTP

One Time Pad.

Overall Diffusion

That property of an ideal block cipher in which a change of even a single message or plaintext bit will change every ciphertext bit with probability 0.5. In practice, a good block cipher will approach this ideal. This means that about half of the output bits should change for any possible change to the input block.

Overall diffusion means that the ciphertext will appear to change at random even between related message blocks, thus hiding message relationships which might be used to attack the cipher.

Overall diffusion can be measured statistically in a realized cipher and used to differentiate between better and worse designs. Overall diffusion does not, by itself, define a good cipher, but it is *required* in a good block cipher.

Also see diffusion, avalanche, strict avalanche criterion and complete.

Padding

In classical cryptography, random data added to the start and end of messages so as to conceal the length of the message, and the position where coding actually starts.

In more conventional computing, some additional data needed to fill-out a fixed-size data structure. This meaning also exists in cryptography, where the last block of a fixed-size block cipher often must be padded to fill the block.

Password

A key, in the form of a word. Also "pass phrase," for multiple-word keys. See: user authentication.

Patent

The legal right, formally granted by a government, to exclude others from making, selling or using the particular invention described in the patent deed. (The term "selling" is generally understood to cover free distribution.) Note that a patent is *not* the right to make the invention, if it is covered by *other* unexpired patents. A patent constitutes the open *publication* of an invention, in return for a limited-term monopoly on its use. A patent is said to protect the *application* of an idea (as opposed to the idea itself), and is distinct from copyright, which protects the *expression* of an idea.

The concept behind patenting is to establish intellectual property in a way somewhat related to a mining claim or real estate. An inventor of a machine or process can file a *claim* on the innovation, provided that it is not previously published, and that someone else does not already have such a claim. Actual patents normally do not claim an overall machine, but just the newly-innovative part, and wherever that part is used, it must be licensed from the inventor. It is common for an inventor to refine earlier work patented by someone else, but if the earlier patent has not expired, the resulting patent often cannot be practiced without a license from the earlier patent holder.

Someone who comes up with a patentable invention and wishes to give up *their* rights can simply publish a full description of the invention. Simple publication should prevent an application from anyone who has not already established legal proof that they previously came up with the same invention. In the U.S., publication also apparently sets a 1-year clock running for an application to be filed by anyone who *does* have such proof. But coming up with an invention does not take away *someone else's* rights if they came up with the same thing first, they may have a year to file, and their case might take several years to prosecute and issue.

In the U.S., a patent is a non-renewable grant, previously lasting 17 years from issue date, now lasting 20 years from application date. Both an application fee and an issue fee are required, as are periodic "maintenance" fees throughout the life of the patent. There are four main requirements:

1. **Statutory Class** (35 USC 101): The invention must be either:
   - a process,
   - a machine,
   - a manufacture,
   - a composition of materials, or
   - a new use for one of the above.
2. **Utility** (35 USC 101): The invention must be of some use.
3. **Novelty** (35 USC 102): The invention must have some aspect which is different from all previous inventions and public knowledge.

   A U.S. patent is **not** available if -- **before the *invention* date** -- the invention was:
   - Publicly known or used the United States of America, or
   - Described in a printed publication (e.g., available at a public library) anywhere

   (35 USC 102(a)).

   A U.S. patent is **not** available if -- **more than a year before the *application* date** -- the invention was:
   - In public use or on sale in the United States of America, or
   - Described in a printed publication (e.g., available at a public library) anywhere

   (35 USC 102(b)).
4. **Unobviousness** (35 USC 103): The invention must have **not** been obvious *to someone of ordinary skill* in the field of the invention *at the time of the invention.* Unobviousness has various general arguments, such as:
   - Unexpected Results,
   - Unappreciated Advantage.
   - Solution of Long-Felt and Unsolved Need, and
   - Contrarian Invention (contrary to teachings of the prior art),

   among many others.

When the same invention is claimed by different inventors, deciding who has "priority" to be awarded the patent can require *legally provable* dates for both "conception" and "reduction to practice":

- *Conception* can be proven by disclosure to others, preferably in documents which can be signed and dated as having been read and understood. The readers can then testify as to exactly what was known and when it was known.
- *Reduction to Practice* may be the patent application itself, or requires others either to watch the invention operate or to make it operate on behalf of the inventor. These events also should be carefully recorded in written documents with signatures and dates.

"In determining priority of invention, there shall be considered not only the respective dates of conception and reduction to practice of the invention, but also the reasonable diligence of one who was first to conceive and last to reduce to practice . . ." (35 USC 102(g)).

Also see: prior art and our claims tutorial.

In practice, a patent is rarely the intrusive prohibitive right that it may at first appear to be, because patents are really about *money* and *respect.* Ideally, a patent rewards the inventor for doing research and development, and then disclosing an invention to the public; it is also a legal recognition of a contribution to society. If someone infringes a patent in a way which affects sales, or which implies that the inventor cannot do anything about it, the patent holder can be expected to show some interest. But when little or no money is involved, a patent can be infringed repeatedly with little or no response, and typically this will have no effect on future legal action.

This simple introduction cannot begin to describe the complexity involved in filing and prosecuting a patent application. Your author does *not* recommend going it alone, unless one is willing to put far more time into learning about it and doing it than one could possibly imagine.

Patent Infringement

Patent infringement occurs when someone makes, sells, or uses a patented invention without license from the patent holder.

Normally the offender will be contacted, and there may be a settlement and proper licensing, or the offender may be able to design around the patent, or offender may simply stop infringing. Should none of these things occur, the appropriate eventual response is a patent infringement lawsuit in federal court.

Perfect Secrecy

The unbreakable strength delivered by a cipher in which all possible ciphertexts may be key-selected with equal probability given any possible plaintext. This means that no ciphertext can imply any particular plaintext any more than any other. This sort of cipher needs as much keying information as there is message information to be protected. A cipher with perfect secrecy has at least as many keys as messages, and may be seen as a (huge) Latin square.

There are some examples:

- (Theoretically) the one-time pad with a perfectly random pad generator.
- The Dynamic Transposition cipher approaches perfect secrecy in that every ciphertext is a bit-permuted balanced block. Thus, every possible plaintext block is just a particular permutation of any ciphertext block. Since the permutation is created by a keyed RNG, we expect any particular permutation to "never" re-occur, and be easily protected from defined plaintext attack with the usual message key. We also expect that the RNG itself will be protected by the vast number of different sequences which could produce the exact same bit-pattern for any ciphertext result.

Also see: ideal secrecy. From Claude Shannon.

Permutation

The mathematical term for a particular arrangement of symbols, objects, or other elements. With *n* symbols, there are

```
P(n) = n*(n-1)*(n-2)*...*2*1 = n!
```

or *n*- factorial possible permutations. The number of permutations of *n* things taken *k* at a time is:

```
P(n,k) = n! / (n-k)!
```

See the permutations section of the Ciphers By Ritter / JavaScript computation pages. Also see combination and symmetric group.

A block cipher can be seen as a transformation between plaintext block values and ciphertext block values, and is thus an emulated simple substitution on huge block-wide values. Both plaintext and ciphertext have the same set of possible block values, and when the ciphertext values have the same ordering as the plaintext, ciphering is obviously ineffective. So *effective* ciphering depends upon *re-arranging* the ciphertext values from the plaintext ordering, which is a *permutation* of the plaintext values. A block cipher is keyed by constructing a *particular* permutation of ciphertext values.

Within an explicit table, an arbitrary permutation (one of the set of all possible permutations) can be produced by shuffling the elements under the control of a random number generator. If, as usual, the random number generator has been initialized from a key, a particular permutation can be produced for each particular key; thus, each key selects a particular permutation.

Also, the second part of substitution-permutation block ciphers: First, substitution operations diffuse information across the width of each substitutions. Next, "permutation" operations act to re-arrange the bits of the substituted result (more clearly described as a set of transpositions); this ends a single round. In subsequent rounds, further substitutions and transpositions occur until the block is thoroughly mixed and overall diffusion hopefully achieved.

PGP

A popular public key cipher system using both RSA and IDEA ciphers. RSA is used to tranfer a random key; IDEA is used to actually protect the message.

One problem with PGP is a relatively unworkable facility for authenticating public keys. While the users can compare a cryptographic hash of a key, this requires communication through a different channel, which is more than most users are willing to do. The result is a system which generally supports man-in-the-middle attacks, and these do **not** require "breaking" either of the ciphers.

Physically Random

A random value or sequence derived from a physical source, typically thermal-electrical noise. Also called really random and truly random.

Pink Noise

A random-like signal in which the magnitude of the spectrum at each frequency is proportional to the inverse of the frequency, or 1/f. At twice the frequency, we have half the energy, which is -3 dB. This is a frequency-response slope of -3 dB / octave, or -10 dB / decade. As opposed to white noise, which has the same energy at all frequencies, pink noise has more low-frequency or "red" components, and so is called "pink."

A common frequency response has half the output voltage at twice the frequency. But this is actually one-quarter the power and so is a -6 dB / octave drop. For pink noise, the desired voltage drop per octave is 0.707.

Plaintext

Plaintext is the original, readable message. It is convenient to think of plaintext as being actual language characters, but may be any other symbols or values (such as arbitrary computer data) which need to be protected.

Poisson Distribution

In statistics, a simplified form of the binomial distribution, justified when we have:

1. a large number of trials *n*,
2. a small probability of success *p*, and
3. an expectation *np* much smaller than SQRT(*n*).

The probability of finding exactly *k* successes when we have expectation *u* is:

```
            k  -u
P(k,u) = u  e    /  k!
```

where *e* is the base of natural logarithms:

```
e = 2.71828...
```

and *u* is:

```
u = n p
```

again for *n* independent trials, when each trial has success probability *p*. In the Poisson distribution, *u* is also both the mean and the variance

The ideal distribution is produced by evaluating the probability function for all possible *k*, from 0 to *n*.

If we have an experiment which we think *should* produce a Poisson distribution, and then repeatedly and systematically find very improbable test values, we may choose to reject the null hypothesis that the experimental distribution is in fact Poisson.

Also see the Poisson section of the Ciphers By Ritter / JavaScript computation pages.

Polyalphabetic Combiner

A combining mechanism in which one input selects a substitution alphabet (or table), and another input selects a value from within the selected alphabet, said value becoming the combining result. Also called a Table Selection Combiner.

Polyalphabetic Substitution

A type of substitution in which multiple distinct simple substitution alphabets are used.

Polygram Substitution

A type of substitution in which one or more symbols are substituted for one or more symbols. The most general possible substitution.

Polygraphic

Greek for "multiple letters." A cipher which translates multiple plaintext symbols at a time into ciphertext. As opposed to monographic; also see homophonic and polyphonic.

Polynomial

Mathematically, an expression in the standard form of:

$$c_n x^n + . . . + c_1 x + c_0$$

The c's or *coefficients* are elements of some field F. The *degree n* is the value of the exponent of the highest power term. A mod 2 polynomial of degree *n* has *n*+1 bits representing the coefficients for each power: *n*, *n*-1, ..., 1, 0.

Perhaps the most insightful part of this is that the addition of coefficients for a particular power does not "carry" into other coefficients or columns.

Polyphonic

Greek for "multiple sounds." The concept of having a letter sequence which is pronounced in distinctly different ways, depending on context. In cryptography, a cipher which uses a single ciphertext symbol to represent multiple different plaintext symbols. Also see homophonic, polygraphic and monographic.

Population

In statistics, the size, or the number of distinct elements in the possibly hidden universe of elements which we can only know by sampling.

Population Estimation

In statistics, techniques used to predict the population based only on information from random samples on that population. See augmented repetitions.

Power

In statistics, the probability of rejecting a false null hypothesis, and thus accepting a true alternative hypothesis.

In DC electronics, simply voltage times current. In AC electronics, the instantaneous product of voltage times current, integrated over a repetitive cycle. In either case the result is in watts, denoted W.

Primitive

A value within a finite field which, when taken to increasing powers, produces all field values except zero. A primitive binary polynomial will be irreducible, but not all irreducibles are necessarily primitive.

Primitive Polynomial

An irreducible polynomial, primitive within a given field, which generates a maximal length sequence in linear feedback shift register (LFSR) applications.

All primitive polynomials are irreducible, but irreducibles are not necessarily primitive, unless the degree of the polynomial is a Mersenne prime. One way to find a primitive polynomial is to select an appropriate Mersenne prime degree and find an irreducible using Algorithm A of Ben Or:

```
1.  Generate a monic random polynomial gx of degree n over GF(q);
2.  ux := x;
3.  for k := 1 to (n DIV 2) do
4.     ux := ux^q mod gx;
5.     if GCD(gx, ux-x) <> 1 then go to 1 fi;
6.     od
```

Ben-Or, M. 1981. Probabilistic algorithms in finite fields. *Proceedings of the 22nd IEEE Foundations of Computer Science Symposium.* 394-398.

The result is a certified irreducible. GF(q) represents the Galois Field to the prime base *q*; for mod 2 polynomials, *q* is 2. These computations require mod 2 polynomial arithmetic operations for polynomials of large degree; "ux^q" is a polynomial squared, and "mod *gx*" is a polynomial division. A "monic" polynomial has a leading coefficient of 1; this is a natural consequence of mod 2 polynomials of any degree. The first step assigns the polynomial "x" to the variable *ux*; the polynomial "x" is x^1, otherwise known as "10".

To get primitives of non-Mersenne prime degree n, we certify irreducibles P of degree n. To do this, we must factor the value 2^n - 1 (which can be a difficult problem, in general). Then, for each factor d of 2^n - 1 we create the polynomial T(d) which is x^d + 1; this is a polynomial with just two bits set: bit d and bit 0. If P evenly divides T(d) for some divisor d, P cannot be primitive. So if P does not divide any T(d) for all distinct divisors d of 2^n - 1, P is primitive.

Prime

In general, a positive integer which is evenly divisible only by itself and 1.

Small primes can be found though the ever-popular Sieve of Eratosthenes, which can also be used to develop a list of small primes used for testing individual values. A potential prime need only be divided by each prime equal to or less than the square-root of the value of interest; if any remainder is zero, the number is not prime.

Large primes can be found by probabilistic tests.

Prior Art

In patents, the knowledge published or otherwise available to the public as of some date. Traditionally, this "knowledge" is in ink-on-paper articles or patents, both of which have provable release dates. Private "in house" journals available only within a company generally would not be prior art, nor would information which has been kept secret. Normally, we expect prior art information to be available in a public library.

In a U.S. application for patent, we are interested in the state of the open or public art as it existed as of the invention date, and also one year prior to the filing date. It is that art -- and not something hidden or something later -- against which the new application must be judged. Many things which seem "obvious" in retrospect were really quite innovative at the time they were done.

PRNG

Pseudo Random Number Generator. In general, *pseudo*randomness is the norm. Any computer random number generator which is not explicitly labeled as physically random, really random, or other such description, is almost certainly *pseudo*random.

Process

In statistics, a sequence of values; a source or generator of such a sequence; a function.

Pseudorandom

A value or sequence of values typically produced by a random number generator, a deterministic computational mechanism. As opposed to really random. Also see random.

The usual random number generator is actually ***pseudo*random**. Given the initial state, the entire subsequent sequence is completely pre-determined, but nevertheless exhibits many of the expected characteristics of a random sequence. Pseudorandomness supports generating the exact same cryptographic sequence repeatedly at different times or locations. Pseudorandomness is generally produced by a mathematical process, which may provide good assurances as to the resulting statistics, assurances which a really random generator generally cannot provide.

Public Key Cipher

Also called an asymmetric cipher or a *two-key* cpher. A cipher which uses one key to encipher a message, and a *different* key to decipher the resulting ciphertext. This allows the enciphering key to be exposed, without exposing the message. As opposed to a secret key cipher.

Either key can be used for enciphering or deciphering. Usually the exposed key is called the "public" key, and the retained hidden key is called the "private" key. The public key is distributed widely, so anyone can use it to encipher a message which presumably can only be deciphered by the hidden private key on the other end. Note that the enciphering end normally does not possess a key which will decipher a message which was just enciphered.

The whole scheme of course depends upon the idea that the private key cannot be developed from knowledge of the public key. The cipher also must resist both known-plaintext *and* defined-plaintext attack (since anyone can generate any amount of plaintext and encipher it). A public key cipher is vastly slower than a secret key cipher, and so is normally used simply to deliver the message key or session key for a conventional or secret key cipher.

Although at first proclaimed as a solution to the key distribution problem, it soon became apparent that someone could *pretend* to be someone else, and send out a "spoofed" public key. When people use that key, the spoofer could receive the message, decipher and read it, then re-encipher the message under the correct key and send it to the correct destination. This is known as a man-in-the-middle (MITM) attack.

A MITM attack is unusual in that it can penetrate cipher security *without* "breaking" *either* the public key cipher *or* the internal secret key cipher, and takes almost no computational effort. This is *extremely serious* because it means that the use of even "unbreakable" ciphers is **not** sufficient to guarantee privacy. All the effort spent on proving the strength of either cipher is simply wasted when a MITM attack is possible, and MITM attacks are only possible with public key ciphers.

To prevent spoofing, public keys must be authenticated (or *validated* or *certified*) as representing who they claim to represent. This can be almost as difficult as the conventional key distribution problem and generally requires complex protocols. And a failure in a key certification protocol can expose a system which uses "unbreakable" ciphers. In contrast, the simple use of an "unbreakable" secret key cipher (with hand-delivered keys) **is** sufficient to guarantee security. This is a real, vital difference between ciphering models.

Random

A process which selects unpredictably, each time independent of all previous times, from among multiple possible results; or a result from such a process. Ideally, an arbitrary stateless selection from among equiprobable outcomes, thus producing a uniform distribution of values. The absence of pattern. Also see pseudorandom.

Randomness is an attribute of the *process* which generates or selects "random" numbers rather than the numbers themselves. But the numbers do carry the ghost of their creation: If values really are randomly generated with the same probability, we expect to find *almost* the same number of occurrences of each value or each sequence of the same length. Over many values and many sequences we expect to see results form in distributions which accord with our understanding of random processes. So if we do not find these expectations in the resulting numbers, we may have reason to suspect that the generating process is not random. Unfortunately, any such suspicion is necessarily statistical in nature, and cannot produce absolute proof in either direction: Randomness can produce *any* relationship between values, including apparent correlations (or their lack) which do not in fact represent the systematic production of the generator. (Also see the discussions of randomness testing in Statistics and Null Hypothesis, the article Chi-Square Bias in Runs-Up/Down RNG Tests, also Randomness Tests: A Literature Survey, in the Literature Surveys and Reviews section of the Ciphers By Ritter page, and Randomness Links, in Ritter's Net Links page.)

From one point of view, there are no "less random" or "more random" sequences, since any sequence can be produced by a random process. And any sequence (at least any *particular* sequence) *also* can be produced by a deterministic computational random number generator. (We note that such generators are specifically designed to and do pass statistical randomness tests.) So the difference is not in the sequences, *per se*, but instead in the generators: For one thing, an RNG sequence is deterministic and therefore may somehow be predicted. But, in practice, extensive analysis could show deviations from randomness in *either* the deterministic RNG designs *or* the nondeterministic really random generation equipment, and this could make even a nondeterministic generator somewhat predictable.

There are "more complex" and "less complex" sequences according to various measures. For example:

- Linear complexity grades sequences on the size of the minimum shift-register state needed to produce the sequence.
- Kolmogorov-Chaitin complexity grades sequences on the size of the description of the algorithm needed to produce the sequence.

These measures produce values related to the amount of pattern in a sequence, or the extent to which a sequence can be predicted by some algorithmic model. Such values describe the uncertainty of a sequence, and are in this way related to entropy.

We should note that the subset of sequences which have a high linear complexity leaves a substantial subset which does not. So if we avoid sequences with low linear complexity, any sequence we do accept must be *more* probable than it would be in the unfiltered set of all possible sequences. In this case, the expected higher uncertainty of the sequence itself is at least partly offset by the certainty that such a sequence will be used. Similar logic applies to S-box measurement and selection.

Oddly -- and much like strength in ciphers -- the "unpredictable" part of randomness is contextual and subjective, rather than the absolute and objective qualities we like in Science. While the sequence from a complex RNG can *appear* random, if we know the secret of the generator construction, and its state, we can predict the sequence exactly. But often we are in the position of seeing the sequence alone, *without* knowing the source, the construction, or the internal state. So while *we* might see a sequence as "random," that same sequence might be absolutely predictable (and thus *not* random) to someone who knows "the secret."

Random Number Generator

A random number generator is a standard computational tool which creates a sequence of apparently unrelated numbers which are often used in statistics and other computations.

In practice, most random number generators are deterministic computational mechanisms, and each number is directly determined from the previous state of the mechanism. Such a sequence is often called pseudo-random, to distinguish it from a really random, sequence somehow composed of actually unrelated values.

A computational random number generator will always generate the same sequence if it is started in the same state. So if we initialize the state from a key, we can use the random number generator to shuffle a table into a particular order which we can reconstruct any time we have the same key. (See, for example: A Keyed Shuffling System for Block Cipher Cryptography.)

Note that random number generators are *designed* to pass the many statistical tests of randomness; clearly, such tests do not indicate a really random sequence. Moreover, if we define "random" as "the absence of any pattern," the only way we could validate such a sequence is by checking for every possible pattern. But there are too many patterns, so "real" randomness would seem to be impossible to check experimentally. (Also see the discussions of randomness testing in Statistics and Null Hypothesis.)

Also see the article: The Efficient Generation of Cryptographic Confusion Sequences, plus RNG Implementations: A Literature Survey, RNG Surveys: A Literature Survey, in the Literature Surveys and Reviews section of the Ciphers By Ritter page, and Randomness Links, in Ritter's Net Links page.

Random Variable

In statistics, a term or label for an unknown value. Also used when each of the possible values have some known probability.

A *discrete* random variable takes on a finite set of values. The probability of each value is the *frequency function* or *probability density function*, and the graph of the frequency function is the frequency distribution.

Range

The set of the results from a mapping for all possible arguments. Also see: domain.

Really Random

A random value or sequence derived from a source which is expected to produce no predictable or repeatable relationship between values.

Examples of a really random source might include radioactive decay, Johnson or thermal noise, shot noise from a Zener diode or reverse-biased junction in breakdown, etc. Clearly, some sort of circuitry will be required to detect these generally low-level events, and the quality of the result is often directly related to the design of the electronic processing. Other sources of randomness might be precise keystroke timing, and the accumulated hash of text of substantial size. Also called physically random and truly random. As opposed to pseudorandom (see random number generator).

Really random values are particularly important as message key objects, or as a sequence for use in a realized one-time pad.

Also see: Random Number Machines: A Literature Survey and Random Electrical Noise: A Literature Survey, in the Literature Surveys and Reviews section of the Ciphers By Ritter page, and Randomness Links, in Ritter's Net Links page.

Relay

Classically, an electro-mechanical component consisting of a mechanical switch operated by the magnetic force produced by an electromagnet, a conductor wound around an iron dowel or core. A relay is at least potentially a sort of mechanical (slow) and nonlinear amplifier which is well-suited to power control.

Research Hypothesis

In statistics, the statement formulated so that the logically contrary statement, the null hypothesis H_0 has a test statistic with a known distribution for the case when there is nothing unusual to detect. Also called the alternative hypothesis H_1, and logically identical to "NOT-H_0" or "H_0 is not true."

Resistor

A basic electronic component in which voltage and current are linearly related by Ohm's Law: **E = IR**. Resistors can thus be used to limit current I given voltage E: (I = E/R), or to produce voltage E from current I: (E = IR). Two resistors in series can divide voltage Ein to produce the output voltage Eo: (Eo = Ein(R1/(R1+R2))).

Also see capacitor and inductor.

Ring

In abstract algebra, a nonempty set R with two dyadic (two-input, one-output) operations which we choose to call "addition" and "multiplication" and denote + and * as usual. If elements (not necessarily numbers) a, b are in R, then a+b is in R, and ab (or a*b) are also in R. The following properties hold:

1. **Addition is commutative:** a + b = b + a
2. **Addition is associative:** (a + b) + c = a + (b + c)
3. **There is a "zero" or additive identity:** a + 0 = a
4. **There is an additive inverse:** for any a there is an x in R such that a + x = 0
5. **Multiplication is associative:** (ab)c = a(bc)
6. **Multiplication is distributive:** a(b + c) = ab + ac and (b + c)a = ba + ca

7. **In a commutative ring, multiplication is commutative:** ab = ba
8. **In a ring with unity, there is a multiplicative identity:** for e in R, ea = ae = a

Root

A solution: A value which, when substituted for a variable in a mathematical equation, makes the statement true.

RMS

root mean square.

Root Mean Square

The square root of the integral of instantaneous values squared. Thus, when measuring voltage or current, a value proportional to the average power in watts, even in a complex waveform.

RNG

Random Number Generator.

Round

In the context of block cipher design, a term often associated with a Feistel block cipher such as DES. A round is the set of operations which are repeated multiple times to produce the final data. For example, DES uses 16 generally identical rounds, each of which performs a number of operations. As opposed to a layer, which is not applied repeatedly.

RSA

The name of an algorithm published by Ron Rivest, Adi Shamir, and Len Adleman (thus, R.S.A.). The first major public key system.

Based on number-theoretic concepts and using huge numerical values, a RSA key must be perhaps ten times or more as long as a secret key for similar security.

Running Key

The confusion sequence in a stream cipher.

Salt

An unnecessarily cute and sadly non-descriptive name for an arbitrary value, unique to a particular computer or installation, prepended to a password before hash authentication. The "salt" acts to complicate attacks on the password user-identification process by giving the same password different hash results on different systems. Ideally, this would be a sort of keying for a secure hash.

Sample

In statistics, one or more elements, typically drawn at random from some population.

Normally, we cannot hope to examine the full population, and so must instead investigate *samples* of the population, with the hope that they represent the larger whole. Often, random sampling occurs "without replacement"; effectively, each individual sample is returned to the population before the next sample is drawn.

S-Box

Substitution box or table; typically a component of a cryptographic system. "S-box" is a rather non-specific term, however, since S-boxes can have more inputs than outputs, or more outputs than inputs, each of which makes a single invertible table impossible. The S-boxes used in DES contain multiple invertible substitution tables, with the particular table used at any time being data-selected.

One possible S-box is the identity transformation (0->0, 1->1, 2->2, ...) which clearly has no effect at all, while every other transformation has at least some effect. So different S-boxes obviously *can* contain different amounts of some qualities. Qualities often mentioned include avalanche and Boolean function nonlinearity. However, one might expect that different ciphering structures will need *different* table characteristics to a greater or less degree. So the discussion of S-box strength always occurs within the context of a particular cipher construction.

S-Box Avalanche

With respect to avalanche, any input change -- even one bit -- will select a different table entry. Over all possible input values and changes, the number of output bits changed will have a binomial distribution. (See the bit changes section of the Ciphers By Ritter / JavaScript computation pages.) So, in this respect, all tables are equal.

On the other hand, it is possible to arrange tables so that single-bit input changes are guaranteed to produce at least two-bit output changes, and this would seem to improve avalanche. But we note that this is *probable* even with a randomly-constructed table, so we have to ask just how much this guarantee has improved things. In a Feistel cipher, it seems like this *might* reduce the number of needed rounds by one. But in actual operation, the plaintext block is generally *randomized*, as in CBC-mode. This means that the probability of getting a single-bit change in operation is very low anyway.

It is true that cipher avalanche is tested using single-bit input changes, and that is the way avalanche is defined. The point of this is to assure that every output bit is "affected" by every input bit. But I see this as more of an experimental requirement than an operational issue that need be optimized.

S-Box Nonlinearity

With respect to Boolean function nonlinearity, as tables get larger it becomes very difficult -- and essentially impossible -- to find tables with ideal nonlinearity values. This means that we are always accepting a compromise value, and this is especially the case if the table must also have high values of other S-box qualities.

Even randomly-constructed tables tend to have reasonable nonlinearity values. We might expect an 8-bit table to have a nonlinearity of about 100 (that is, 100 bits must change in one of the eight 256-bit output functions to reach the closest affine Boolean function). Experimental measurement of the nonlinearity of 1,000,000 random 8-bit tables shows exactly one table with a nonlinearity as low as 78, and the computed probability of an actually *linear* table (nonlinearity zero) is something like 10^{-72} or 2^{-242}.

The NSA-designed 8-bit table in Skipjack cipher has a computed nonlinearity of 104. While not quite the highest value we could find, it *is* in the top 2.5 percent of the distribution, and it seems improbable that this occurred by accident. We might assume that this table is representative of the modern understanding of the needs of a Feistel design with a fixed table. If so, we might conclude that good nonlinearity (or something very much like it) is a necessary, if not quite sufficient, part of the design.

Keyed S-Boxes

It is "easy" to construct keyed S-boxes, by shuffling under the control of a keyed cryptographic random number generator. (See, for example: A Keyed Shuffling System for Block Cipher Cryptography.) This has the significant advantage of providing no fixed tables for The Opponent to understand and attack.

One question is whether one should attempt to measure and discard tables with poorer qualities than others. My personal feeling is that the ciphering structure should be strong enough to handle the expected random table distribution without added measurement and selection.

Also see: S-Box Design: A Literature Survey, in the Literature Surveys and Reviews section of the Ciphers By Ritter page.

Scalable

A cipher design which can produce both large real ciphers and tiny experimental versions from the exact same construction rules. Scalability is about more than just variable size: Scalability is about establishing a uniform structural identity which is size-independent, so that we achieve a strong cipher near the top, and a tiny but accurate model that we can investigate near the bottom.

While full-size ciphers can never be exhaustively tested, tiny cipher models *can* be approached experimentally, and any flaws in them probably will be present in the full-scale versions we propose to use. Just as mathematics works the same for numbers large or small, a backdoor cipher built from fixed construction rules must have the same sort of backdoor, whether built large or small.

For block ciphers, the real block size must be at least 128 bits, and the experimental block size probably should be between 8 and 16 bits. Such tiny ciphers can be directly compared to keyed substitution tables of the same size, which are the ideal theoretical model of a block cipher.

Potentially, scalability does far more than just *simplify* testing: Scalability is an enabling technology that supports experimental analysis which is otherwise *impossible.*

Secrecy

One of the objectives of cryptography: Keeping private information private. Also see: trust.

In a secret key cipher, secrecy implies the use of a strong cipher. Secrecy in communication requires the secure distribution of secret keys to both ends (this is the key distribution problem).

In a public key cipher, the ability to expose keys apparently solves the key distribution problem. But communications secrecy requires that public keys be authenticated (*certified*) as belonging to their supposed owner. This must occur to cryptographic levels of assurance, because failure leads to immediate vulnerability under a man-in-the-middle attack. The possibility of this sort of attack

is very disturbing, because it needs little computation, and does not involve breaking any cipher, which makes all discussion of cipher strength simply irrelevant.

Secret Code

A coding in which the correspondence between symbol and code value is kept secret.

Secret Key Cipher

Also called a symmetric cipher or conventional cipher. A cipher in which the exact same key is used to encipher a message, and then decipher the resulting ciphertext. As opposed to a public key cipher.

Security

Protection of a vital quality (such as secrecy, or safety, or even wealth) from infringement, and the resulting relief from fear and anxiety. The ability to engage and defeat attempts to damage, weaken, or destroy a vital quality. Security, in the form of assuring the secrecy of information while in storage or transit, is the fundamental role of cryptography.

A secure cryptosystem physically or logically *prevents* unauthorized disclosure of its protected data. This is *independent* of whether the attacker is a government agent, a criminal, a private detective, some corporate security person, or a friend of an ex-lover. Real security does not care *who* the attacker is, or *what* their motive may be, but instead protects against the threat itself. Limited security, on the other hand, often seeks to guess the identity, capabilities and motives of the attacker, and concentrates resources at those points.

There is, of course, no *absolute* security. But we can have real security against particular, defined threats. Also see: strength.

Security Through Obscurity

A phrase which normally refers to inventing a new cipher which is supposedly strong, then keeping the cipher secret so it "cannot be attacked." One problem with this strategy is that it prevents public review of the cipher design, which means that the cipher may have serious weaknesses. And it may be much easier for The Opponent to obtain the supposedly secret ciphering program than it would be to break a serious cipher (see Kerckhoff's second requirement).

On the other hand, it can be a mistake to use even a public and well-reviewed cipher, if the cipher protects enough valuable information to support a substantial investment in analysis and equipment to break the cipher. A reasonable alternative is to select from among a wide variety of conceptually different ciphers, each of which thus carries far less information of far less value and so may not warrant a substantial attack investment.

Semiconductor

A material which is between conductor and insulator with respect to ease of electron flow. The obvious examples are silicon and germanium.

Semigroup

A set with an associative dyadic operation which happens to be closed.

Session Key

A key which lasts for the period of a work "session." A message key used for multiple messages.

Set

A collection of distinguishable elements, usually, but not necessarily, numbers.

Shift Register

An array of storage elements in which the values in each element may be "shifted" into an adjacent element. (A new value is shifted into the "first" element, and the value in the "last" element is normally lost, or perhaps captured off-chip.) (See LFSR.)

```
Right-Shifting Shift Register (SR)

            +----+  +----+         +----+
Carry In -->| A0 |->| A1 |-> ... ->| An |--> Carry Out
            +----+  +----+         +----+
```

In digital hardware versions, elements are generally bits, and the stored values actually move from element to element in response to a clock. Analog hardware versions include the charge-coupled devices (CCD's) used in cameras, where the analog values from lines of sensors are sampled in parallel, then serialized and stepped off the chip to be digitized and processed.

In software versions, elements are often bytes or larger values, and the values may not actually move during stepping. Instead, the values may reside in a circular array, and one or more offsets into that array may step. In this way, even huge amounts of state can be "shifted" by changing a single index or pointer.

Shuffle

Generally, the concept of "mixing up" a set of objects, symbols or elements, as in shuffling cards. Mathematically, each possible arrangement of elements is a particular permutation.

Within a computer environment, it is easy to shuffle an arbitrary number of symbols using a random number generator, and the algorithm of Durstenfeld, which is described in Knuth II:

Durstenfeld, R. 1964. Algorithm 235, Random Permutation, Procedure SHUFFLE. *Communications of the ACM.* 7: 420.

Knuth, D. 1981. *The Art of Computer Programming,* Vol. 2, Seminumerical Algorithms. 2nd ed. 139. Reading, Mass: Addison-Wesley.

Sieve of Eratosthenes

A way to find relatively small primes. Although small primes are less commonly useful in cryptography than large (say, 100+ digit) primes, they *can* at least help to validate implementations of the procedures used to find large primes.

Basically, the "Sieve of Eratosthenes" starts out with a table of numbers from 1 to some limit, all of which are potential primes, and the knowledge that 2 is a prime. Since 2 is a prime, no other prime can have 2 as a factor, so we run though the table discarding all multiples of 2. The next remaining number above 2 is 3, which we accept as a prime, and then run through the table crossing off all multiples of 3. The next remaining is 5, so we cross off all multiples of 5, and so on. After we cross-off each prime up to the square-root of the highest value in the table, the table will contain only primes.

A similar process works with small polynomials, and small polynomial fields, to find irreducible polynomials.

Significance

In statistics, the probability of committing a type I error, the rejection of a true null hypothesis. Given the probability distribution of the test statistic for the case "nothing unusual found," that area which is sufficiently unlikely that values in this *critical region* would lead to rejecting the null hypothesis, and thus accepting the alternative hypothesis.

Simple Substitution

A type of substitution in which each possible symbol is given a unique replacement symbol.

Perhaps the original classical form of cipher, in which each plaintext character is enciphered as some different character. In essence, the order of the alphabet is scrambled or permuted, and the particular scrambled order (or the scrambling process which creates that particular order) is the cipher key. Normally we think of scrambling alphabetic letters, but any computer coding can be scrambled similarly.

Small, practical examples of simple substitution are easily realized in hardware or software. In software, we can have a table of values each of which can be indexed or selected by element number. In hardware, we can simply have addressable memory. Given an index value, we can select the element at the index location, and read or change the value of the selected element.

A substitution table will be initialized to contain exactly one occurrence of each possible symbol or character. This allows enciphering to be reversed and the ciphertext deciphered. For example, suppose we substitute a two-bit quantity, thus a value 0..3, in a particular table as follows:

```
2  3  1  0.
```

The above substitution table takes an input value to an output value by selecting a particular element. For example, an input of 0 selects 2 for output, and an input of 2 selects 1. If this is our enciphering, we can decipher with an inverse table. Since 0 is enciphered as 2, 2 must be deciphered as 0, and since 2 is enciphered as 1, 1 must be deciphered as 2, with the whole table as follows:

```
3  2  0  1.
```

Mathematically, a simple substitution is a mapping (from input to output) which is one-to-one and onto, and is therefore invertible.

Software

The description of a logic machine. The original textual composition is called source code, the file of compiled opcode values is called object code, and the final linked result is pure "machine code" or machine language Note that, by itself, software does not and can not function; but instead relies upon hardware for all functionality. When "software" is running, there is no software there: there is only hardware memory, with hardware bits which can be sensed and stored, hardware counters and registers, and hardware digital logic to make decisions. See: computer, system, system design, and debug.

Source Code

The textual representation of a computer program as it is written by a programmer. Nowadays, source is typically in a high-level language like C, C++ or Pascal, but inevitably some programmers must work "close to the machine" in assembly language. The "code" part of this is presumably an extension of the idea that, ultimately, all computer programs are executed as "machine code" or machine language. This consists of numeric values or "operation codes" ("opcodes") which select the instruction to be executed, and so represent a very public *code* for those instructions. Also see object code.

State

Information storage, or "memory." In abstract machine theory, retained information, generally used to influence future events.

In statistics, the current symbol from a sequence, or a value which selects or conditions possible outcomes (see: Markov process).

We normally measure "state" in units of information or bits, and 8 bits of "state" can support 2^8 or 256 different state-value combinations or *states*.

Also see: deterministic and keyspace.

Stationary Process

In statistics, a stochastic (random) process (function) whose general statistics do not change over time; in which every sub-sequence is representative of the whole; a homogenous process. This may not be true of a Markov process. Also see: ergodic.

Statistic

A computation or process intended to reduce diverse results into a one-dimensional ordering of values for better understanding and comparison. Also the value result of such a computation. See statistics.

A useful statistic will have some known (or at least explorable) probability distribution for the case "nothing unusual found." This allows the statistic value to be interpreted as the probability of finding that value or less, for the case "nothing unusual found." Then, if improbable statistic values occur repeatedly and systematically, we can infer that something unusual *is* being found, leading to the rejection of the null hypothesis.

It is also possible to explore different distributions for the same statistic under different conditions. This can provide a way to guess which condition was in force when the data were obtained.

Statistics

The mathematical science of interpreting probability to extract meaning from diverse results. Also the analysis of a large population based on a limited number of random samples from that population; this is also the ability to state probability bounds for the correctness of certain types of inductive reasoning. See statistic and random variable.

The usual role of statistics is to identify particular systematic events in the context of expected random variations that may conceal such events. This often occurs in a context of difficult and costly experimentation, and there is a premium on results which are so good that they stand above the noise; it may be that not much is lost if a weak positive is ignored.

In contrast, cryptography and randomness generally support vast amounts of testing at low cost, and we seek weak indications. In this context, we may find it more useful to conduct many tests and collect many statistic values, then visually and mathematically compare the experimental distribution to the ideal for that statistic.

A statistical distribution usually represents what we should expect from random data or random sampling. If we have random data, statistic values exceeding 95% of the distribution (often called *failure*) *should* occur about 1 time in 20. And since that one time may happen on the very first test, it is only prudent to conduct many tests and accumulate results which are more likely to represent reality than any one result from a single test.

In statistical randomness testing, "failure" should and *must* occur with the appropriate frequency. Thus, the failure to fail is itself a failure! This means that the very concept of statistical "failure" often may be inappropriate for cryptographic use. Grading a result as "pass" or "fail" discards all but one bit of information. Further, a pass / fail result is a Bernoulli trial, which would take many, many similar tests to properly characterize. So it may be more appropriate to collect 20 or more statistic probability values, and then compare the accumulation to the expected distribution for that statistic. This will provide a substantial basis for asserting that the sampled process either did or did not produce the same statistic distribution as a random process.

Due to random sampling, any statistical result is necessarily a *probability*, rather than certainty. An "unlucky" sampling can produce statistical results which imply the opposite of reality. In general, statistics simply **cannot** provide the 100 percent certainty which is traditionally expected of mathematical "proof."

Steganography

Greek for "sheltered writing." Methods of cryptology which seek to conceal the *existence* of a message. As opposed to cryptography which seeks to hide the *information* in the message, even if the message itself is completely exposed.

Stochastic

In statistics, random; involving a random variable.

Stream Cipher

A cipher which directly handles messages of arbitrary size, by ciphering individual elements, such as bits or bytes. This avoids the need to accumulate data into a block before ciphering, as is necessary in a conventional block cipher. But note that a stream cipher can be seen as an operating mode, a "streaming" of a tiny block transformation. Stream ciphers can be called "combiner-style" ciphers.

Stream Cipher Diffusion

In a conventional stream cipher, each element (for example, each byte) of the message is ciphered independently, and does not affect any other element.

In a few stream cipher designs, the value of one message byte may change the enciphering of *subsequent* message bytes; this is forward data diffusion. But a stream cipher **cannot** change the enciphering of *previous* message bytes. In contrast, changing even the last bit in a block cipher block will generally change about half of the earlier bits within that same block. Changing a bit in one block may even affect later blocks if we have some sort of stream meta-cipher composed of block cipher transformations, like CBC.

Note that a stream cipher generally does not need data diffusion for strength, as does a block cipher. In a block cipher, it may be possible to separate individual components of the cipher if their separate effects are not hidden by diffusion. But a stream cipher generally re-uses the same transformation, and has no multiple data components to hide.

Stream Cipher Construction

The classic stream cipher is very simple, consisting of a keyed random number generator which produces a random-like confusion sequence or running key. That sequence is then combined with plaintext data in a simple additive combiner to produce ciphertext.

When an exclusive-OR combiner is used, exactly the same construction will also decipher the ciphertext. But if The Opponents have some known-plaintext and associated ciphertext, they can easily produce the original confusion sequence. This, along with their expected knowledge of the cipher design, may allow them to attack and expose the confusion generator. If this is successful, it will, of course, break the system until the RNG is re-keyed.

The ultimate stream cipher is the one-time pad, in which a really random sequence is never re-used. But if a sequence *is* re-used, The Opponent can generally combine the two ciphertexts, eliminating the confusion sequence, and producing the combined result of two plaintexts. Such a combination is normally easy to attack and penetrate.

The re-use of confusion sequence is extremely dangerous in a stream cipher design. In general, all stream cipher designs *must* use a message key to assure that the cipher is keyed with a random value for every new ciphering. This does, of course, expand the ciphertext by the size of the message key.

Another alternative in stream cipher design is to use a stronger combiner, such as Latin square or Dynamic Substitution combining. This can drastically reduce the complexity required in the confusion generator, which normally provides all stream cipher strength. Each of these stronger combiners is nonlinear, with substantial internal state, and the designer may elect to use multiple combinings in sequence, or a selection among different combiners. Neither of these approaches make much sense with an additive combiner.

Strength

The ability of a cipher to resist attack and maintain secrecy. The overall "strength" of a cipher is the minimum effort required to break the cipher, by any possible attack. But our *knowledge* of cipher "strength" is necessarily contextual and subjective, much like *unpredictability* in random sequences. Although "strength" would seem to be the entire point of using a cipher, cryptography

has no way to measure strength.

Cipher "strength" is often taken as an absolute universal *negative*, the simple *non-existence* of any attack which could succeed, assuming some level of attack resources. But this means that overall "strength" may be forever impossible to measure, because there is no hope of enumerating and evaluating *every possible* attack.

Strength and Cryptanalysis

Because we have no tools for the discussion of strength under all possible attacks, cipher "strength" is normally discussed in the context of particular attacks. Each known attack approach can be elaborated for a particular cipher, and a value calculated for the effort required to break the cipher in that way; this may set an "upper bound" on the unknown strength of the cipher (although some "elaborations" are clearly better than others). And while this is certainly better than not knowing the strength with respect to known attacks, such attacks may not represent the actual threat to the cipher in the field. (A cipher may even be said to have *different* "contextual strengths," depending on the knowledge available to different Opponents.) In general, we never know the "lower bound" or "true" strength of a cipher. So, unless a cipher is shown to be weaker than we can accept, cryptanalysis provides no useful information about cipher strength.

It is sometimes argued that "our guys" are just as good as the Opponents, who thus could not break a cipher with less effort than we know. Or it is said that if a better break were known, that secret necessarily would get out. When viewed in isolation such statements are clearly false reasoning, yet these are the sort of assumptions that are often implicitly used to assert strength after cryptanalysis.

Since we cannot know the true situation, for a proper security analysis we must instead assume that our Opponents have more time, are better trained, are better equipped, and may even be *smarter* than our guys. Further, the Opponents are quite likely to function as a well-motivated group with a common goal and which can keep secrets; clearly, this is a far different situation than the usual academic cryptanalysis. So, again, cryptanalysis by "our guys" provides **no** information about the strength of the cipher as seen by our Opponents.

Increasing Probable Strength and Reducing Possible Loss

Technical strength is just one of the many possibilities for weakness in a cipher system, and perhaps even the least likely. It is surprisingly difficult to construct a cipher system without "holes," despite using good ciphers, and The Opponents get to exploit any overlooked problems. Users must be educated in security, and must actively keep secrets or there will be nothing to protect. In contrast, cryptanalysis is very expensive, success is never assured, and even many of the known attacks are essentially impossible in practice.

Nevertheless, it is a disturbing fact that we do not know and cannot guarantee a "true" strength for any cipher. But there *are* approaches which may reduce the probability of technical weakness and the extent of any loss:

1. We can *extrapolate* various attacks beyond weakness levels actually shown, and thus possibly avoid some weak ciphers.
2. We can use systems that change ciphers periodically. This will reduce the amount of information under any one cipher, and so limit the damage if that cipher is weak.
3. We can use multiple encryption with different keys and different ciphers as our standard mode. In this way, not just one but multiple ciphers must each be penetrated simultaneously to expose the protected data.
4. We can use systems that allow us to stop using ciphers when they are shown weak, and switch to others.

Kinds of Cipher Strength

In general, we can consider a cipher to be a large key-selected transformation between plaintext and ciphertext, with two main types of strength:

- One type of "strength" is an inability to extrapolate from known parts of the transformation (e.g., known plaintext) to model -- or even approximate -- the transformation at new points of interest (message ciphertexts).
- Another type of "strength" is an inability to develop a particular key, given the known cipher and a large number of known transformation points.

Views of Strength

Strength is the effectiveness of fixed defense in the cryptography war. In real war, a strong defense might be a fortification at the top of a mountain which could only be approached on a single long and narrow path. Unfortunately, in real military action, time after time, making assumptions about what the opponent "could not" do turned out to be deadly mistakes. In cryptography we can at least imagine that someday we might *prove* that all approaches but one are actually *impossible*, and then guard that last approach; see mathematical cryptography.

The Future of Strength

It is sometimes convenient to see security as a fence around a restricted compound: We can beef up the front gate, and in some way measure that increase in "strength." But none of that matters if someone cuts through elsewhere, or tunnels under, or jumps over. Until we can produce a cipher design which reduces all the possible avenues of attack to exactly one, it will be very difficult to measure "strength."

One possibility might be to construct ciphers in layers of different puzzles: Now, the obvious point of having multiple puzzles is to require multiple solutions before the cipher is broken. But a perhaps less obvious point is to set up the design so that the solution to one puzzle requires The Opponent to *commit* (in an information sense) in a way that *prevents* the solution to the next puzzle.

Also see design strength, perfect secrecy, ideal secrecy, and security.

Strict Avalanche Criterion (SAC)

A term used in S-box analysis to describe the contents of an invertible substitution or, equivalently, a block cipher. If we have some input value, and then change one bit in that value, we expect about half the output bits to change; this is the avalanche effect, and is caused by an avalanche process. The *Strict Avalanche Criterion* requires that each output bit change with probability one-half (over all possible input starting values). This is stricter than avalanche, since if a *particular* half of the output bits changed *all* the time, a strict interpretationist might call *that* "avalanche." Also see complete.

As introduced in Webster and Tavares:

> "If a cryptographic function is to satisfy the strict avalanche criterion, then each output bit should change with a probability of one half whenever a single input bit is complemented." [p.524]

Webster, A. and S. Tavares. 1985. On the Design of S-Boxes. *Advances in Cryptology -- CRYPTO '85*. 523-534.

Although the SAC has tightened the understanding of "avalanche," even SAC can be taken too literally. Consider the scaled-down block cipher model of a small invertible keyed substitution table: Any input bit-change thus selects a different table element, and so produces a random new value (over all possible keys). But when we compare the new value with the old, we find that typically half the bits change, and sometimes *all* the bits change, but *never* is there no change at all. This is a tiny bias toward change.

If we have a 2-bit (4-element) table, there are 4 values, but after we take one as the original, there are only 3 *changed* values, not 4. We will see changes of 1 bit, 1 bit, and 2 bits. But this is a change expectation of 2/3 for each output bit, instead of exactly 1/2 as one might interpret from SAC. Although this bias is clearly size-related, its source is *invertibility* and the definition of *change*. Thus, even a large block cipher *must* have *some* bias, though it is unlikely that we could measure enough cases to see it. The point is that one can extend some of these definitions well beyond their intended role.

Subjective

In the study of logic, a particular *interpretation* of reality, rather than objective reality itself.

Substitution

The concept of replacing one symbol with another symbol. This might be as simple as a grade-school lined sheet with the alphabet down the left side, and a substitute listed for each letter. In computer science this might be a simple array of values, any one of which can be selected by indexing from the start of the array. See substitution table.

Cryptography recognizes four types of substitution:

- Simple Substitution or Monoalphabetic Substitution,
- Homophonic Substitution,
- Polyalphabetic Substitution, and
- Polygram Substitution.

Substitution-Permutation

A method of constructing block ciphers in which block elements are substituted, and the resulting bits typically transposed or scrambled into a new arrangement. This would be one round of

many.

One of the advantages of S-P construction is that the "permutation" stage can be simply a re-arrangement of wires, taking almost no time. Such a stage is more clearly described as a limited set of "transpositions," rather than the more general "permutation" term. Since substitutions are *also* permutations (albeit with completely different costs and effects), one might fairly describe such a cipher as a "permutation-permutation cipher," which is not particularly helpful.

A disadvantage of the S-P construction is the need for special substitution patterns which support diffusion. S-P ciphers diffuse bit-changes across the block round-by-round; if one of the substitution table output bits does not change, then no change can be conducted to one of the tables in the next round, which has the effect of reducing the complexity of the cipher. Consequently, special tables are required in S-P designs, but even special tables can only reduce and not eliminate the effect. See Complete.

Substitution Table

(Also S-box.) A linear array of values, indexed by position, which includes any value at most once. In cryptographic service, we normally use binary-power invertible tables with the same input and output range. For example, a byte-substitution table will have 256 elements, and will contain each of the values 0..255 exactly once. Any value 0..255 into that table will select some element for output which will also be in the range 0..255.

For the same range of input and output values, two invertible substitution tables differ only in the order or permutation of the values in the table. There are 256 factorial different byte-substitution tables, which is a keyspace of 1648 bits.

A keyed simple substitution table of sufficient size is the ideal block cipher. Unfortunately, with 128-bit blocks being the modern minimum for strength, there would be 2^{128} entries in that table, which is completely out of the question.

A keyed substitution table of *practical* size can only be thought of as a weak block cipher by itself, but it can be part of a combination of components which produce a stronger cipher. And since an invertible substitution table is the ideal tiny block cipher, it can be used for direct experimental comparison to a scalable block cipher of that same tiny size.

Superencryption

Usually the outer-level encryption of a multiple encryption. Often relatively weak, relying upon the text randomization effect of the lower-level encryption.

Surjective

Onto. A mapping f: *X* -> *Y* where *f(x)* covers all elements in *Y*. Not necessarily invertible, since multiple elements *x* in *X* could produce the same *f(x)* in *Y*.

Switch

Classically, an electro-mechanical device which physically presses two conductors together at a contact point, thus "making" a circuit, and also pulls the conductors apart, thus allowing air to insulate them and thus "breaking" the circuit. More generally, something which exhibits a significant change in some parameter between "ON" and "OFF."

Switching Function

A logic function.

Symmetric Cipher

A secret key cipher.

Symmetric Group

The symmetric group is the set of all one-to-one mappings from a set into itself. The collection of all permutations of some set.

Suppose we consider a block cipher to be a key-selected permutation of the block values: One question of interest is whether our cipher construction could, if necessary, reach every possible permutation, the symmetric group.

System

An interconnecting network of components which coordinate to perform a larger function. Also a system of ideas. See system design.

System Design

The design of potentially complex systems.

It is now easy to construct large hardware or software systems which are almost unmanageably complex and never error-free. But a good design and development approach can produce systems with far fewer problems. One such approach is:

1. Decompose the system into small, *testable* components.
2. Construct and then *actually test* each of the components individually.

This is both easier and harder than it looks: there are *many* ways to decompose a large system, and finding an effective and efficient decomposition can take both experience and trial-and-error. But many of the possible decompositions define components which are less testable or even *un*testable, so the testability criterion greatly reduces the search.

Testing is no panacea: we cannot hope to find all possible bugs this way. But in practice we *can* hope to find 90 percent or more of the bugs simply by *actually testing* each component. (Component testing means that we are forced to *think* about what each component does, and about its requirements and limits. Then we have to make the realized component *conform* to those tests, which were based on our theoretical concepts. This will often expose problems, whether in the implementation, the tests, or the concepts.) By testing all components, when we put the system together, we can hope to avoid having to debug multiple independent problems simultaneously.

Other important system design concepts include:

- Build in test points and switches to facilitate run-time inspection, control, and analysis.
- Use repeatable comprehensive tests at all levels, and when a component is "fixed," run those tests again.
- Start with the most basic system and fewest components, make that "work" (pass appropriate system tests), then "add features" one-by-one. Try not to get too far before making the expanded system work again.

Table Selection Combiner

A combining mechanism in which one input selects a table or substitution alphabet, and another input selects a value from within the selected table, said value becoming the combined result. Also called a Polyalphabetic Combiner.

TEMPEST

Supposedly the acronym for "Transient Electromagnetic Pulse Emanation Surveillance Technology." Originally, the potential insecurity due to the electromagnetic radiation which inherently occurs when a current flow changes in a conductor. Thus, pulses from digital circuitry might be picked up by a receiver, and the plaintext data reconstructed. The general concept can be extended to the idea that plaintext data pulses may escape on power lines, or as a faint background signal to encrypted data, or in any other unexpected electronic way.

Some amount of current change seems inevitable when switching occurs, and modern digital computation is based on such switching. But the amount of electromagnetic radiation emitted depends upon the amount of current switched, the length of the conductor, and the speed of the switching (that is, dI/dt, or the rate-of-change in current). In normal processing the amount of radiated energy is very small, but the value can be much larger when fast power drivers are used to send signals across cables of some length. This typically results in broadband noise which can be sensed with a shortwave receiver, a television, or an AM portable radio. Such receivers can be used to monitor attempts at improving the shielding.

Ideally, equipment would be fully enclosed in an electrically unbroken conducting surface. In practice, the conductive enclosure may be sheet metal or screening, with holes for shielded cables. Shielding occurs not primarily from metal *per se*, but instead from the flow of electrical current in that metal. When an electromagnetic wave passes through a conductive surface, it induces a current, and that current change creates a similar but *opposing* electromagnetic wave which nearly *cancels* the original. The metallic surface must conduct in all directions to properly neutralize waves at every location and from every direction.

Stock computer enclosures often have huge unshielded openings which are hidden by a plastic cover. These should be covered with metal plates or screening, making sure that good electrical contact occurs at all places around the edges. Note that assuring good electrical connections can be difficult with aluminum, which naturally forms a thin but hard and non-conductive surface oxide. It is important to actually monitor emission levels with receivers both before and after any change, and extreme success can be very difficult. We can at least make sure that the shielding is tight (that it electrically conducts to all the surrounding metal), that it is as complete as possible, and that external cables are effectively shielded.

Cable shielding extends the conductive envelope around signal wires and into the envelope surrounding the equipment the wire goes to. Any electromagnetic radiation from within a shield will tend to produce an opposing current in the shield conductor which will "cancel" the original radiation. But if a cable shield is *not* connected at *both* ends, no opposing current can flow, and no electromagnetic shielding will occur, despite having a metallic "shield" around the cable. It is thus necessary to assure that each external cable *has* a shield, and that the shield is *connected* to a conductive enclosure at *both* ends. (Note that some equipment may have an isolating capacitor between the shield and chassis ground to minimize "ground loop" effects when the equipment at each end of the cable connects to different AC sockets.) When shielding is impossible, it can be useful to place ferrite beads or rings around cables to promote a balanced and therefore essentially

non-radiating signal flow.

Perhaps the most worrisome emitter on a personal computer is the display cathode ray tube (CRT). Here we have a bundle of three electron beams, serially modulated, with reasonable current, switching quickly, and repeatedly tracing the exact same picture typically 60 times a second. This produces a recognizable substantial signal, and the repetition allows each display point to be compared across many different receptions, thus removing noise and increasing the effective range of the unintended communication. All things being equal, a liquid-crystal display should radiate a far smaller and also more-complex signal than a desktop CRT.

Transformer

A passive electrical component composed of magnetically-coupled coils of wire. When AC flows through one coil or "primary," it creates a changing magnetic field which induces power in another coil. A transformer thus *isolates* power or signal, and also can change the voltage-to-current ratio, for example to "step down" line voltage for low-voltage use, or to "step up" low voltages for high-voltage devices (such as tubes or plasma devices).

Transistor

An active semiconductor component which performs analog amplification.

Originally, a bipolar version with three terminals: Emitter (e), Collector (c), and Base (b). Current flow through the base-emitter junction (I_{be}) is amplified by the current gain or beta (B) of the device in allowing current to flow through the collector-base junction and on through the emitter (I_{ce}).

In a sense, a bipolar transistor consists of two back-to-back diodes: the base-collector junction (operated in reverse bias) and the base-emitter junction (operated in forward bias) which influence each other. Current through the base-emitter junction releases either electrons or "holes" which are then drawn to the collector junction by the higher potential there, thus increasing collector current. The current ratio between the base input and the collector output is amplification.

Field-Effect Transistors (FET's, as in MOSFET, etc.) have an extremely high input impedence, taking essentially no input current, and may be more easily fabricated in integrated circuits than bipolars. In an FET, Drain (d) and Source (s) contacts connect to a "doped" semiconductor channel. Extremely close to that channel, but still insulated from it, is a conductive area connected to a Gate (g) contact. Voltage on the gate creates an electrostatic field which interacts with current flowing in the drain-source channel, and can act to turn that current ON or OFF, depending on channel material (P or N), doping (enhancement or depletion), and gate polarity. Sometimes the drain and source terminals are interchangeable, and sometimes the source is connected to the substrate. Instead of an insulated gate, we can also have a reverse-biased diode junction, as in a JFET.

N-channel FET's generally work better than p-channel devices. JFET's can only have "depletion mode," which means that, with the gate grounded to the source, they are ON. N-channel JFET devices go OFF with a negative voltage on the gate. Normally, MOSFET devices are "enhancement mode" and are OFF with their gate grounded. N-channel MOSFET devices go ON with a positive voltage (0.5 to 5v) on the gate. Depletion mode n-channel MOSFET devices are possible, but not common.

Transposition

The exchange in position of two elements. The most primitive possible permutation or re-ordering of elements. Any possible permutation can be constructed from a sequence of transpositions.

Trap Door

A cipher design feature, presumably planned, which allows the apparent strength of the design to be easily avoided by those who know the trick. Similar to back door.

Triple DES

The particular block cipher which is the U.S. Data Encryption Standard or DES, performed three times, with two or three different keys.

Truly Random

A random value or sequence derived from a physical source. Also called really random and physically random.

Trust

The assumption of a particular outcome in a dependence upon someone else. Trust is the basis for communications secrecy: While secrecy can involve keeping one's own secrets, *communications secrecy* almost inevitably involves at least a second party. We thus necessarily "trust" that party with the secret itself, to say nothing of cryptographic keys. It makes little sense to talk about secrecy in the absence of trust.

In a true security sense, it is impossible to fully trust *anyone:* Everyone has their weaknesses, their oversights, their own agendas. But normally "trust" involves some form of *commitment* by the other party to keep any secrets that occur. Normally the other party is constrained in some way, either by their own self-interest, or by contractual, legal, or other consequences of the failure of trust. The idea that there can be any realistic trust between two people who have never met, are not related, have no close friends in common, are not in the same employ, and are not contractually bound, can be a very dangerous delusion. It is important to recognize that no trust is without limit, and those limits are precisely the commitment of the other party, bolstered by the consequences of betrayal. Trust without consequences is necessarily a very weak trust.

Truth Table

Typically, a Boolean function expressed as the table of the value it will produce for each possible combination of input values.

Type I Error

In statistics, the rejection of a true null hypothesis.

Type II Error

In statistics, the acceptance of a false null hypothesis.

Unary

From the Latin for "one kind." Sometimes used to describe functions with a single argument, such as "the unary -" (the minus-sign), as opposed to subtraction, which presumably would be "binary," and *that* could get very confusing very fast. Thus, monadic may be a better choice. Also see: binary and dyadic.

Unexpected Distance

The values computed by a fast Walsh transform when calculating Boolean function nonlinearity as often used in S-box analysis.

Given any two random Boolean sequences of the same length, we "expect" to find about half of the bits the same, and about half different. This means that the *expected* Hamming distance between two sequences is half their length.

With respect to Boolean function nonlinearity, the expected distance is not only what we *expect*, it is also *the best we can possibly do*, because each affine Boolean function comes in both complemented and uncomplemented versions. So if *more* than half the bits differ between a random function and one version, then *less* than half must differ to the other version. This makes the expected distance the ideal reference point for nonlinearity.

Since the FWT automatically produces the difference between the expected distance and the distance to each possible affine Boolean function (of the given length), I call this the *un*expected distance. Each term is positive or negative, depending on which version is more correlated to the given sequence, and the absolute value of this is a measure of linearity. But since we generally want *non*linearity, we typically subtract the unexpected value from half the length of the sequence.

Unicity Distance

The amount of ciphertext needed to uniquely identify the correct key and its associated plaintext (assuming a ciphertext-only attack and natural language plaintext). With less ciphertext than the unicity distance, multiple keys may produce decipherings which are each plausible messages, although only one of these would be the correct solution. As we increase the amount of ciphertext, many formerly-plausable keys are eliminated, because the plaintext they produce becomes identifiably different from the structure and redundancy we expect in a natural language.

> "If a secrecy system with a finite key is used, and *N* letters of cryptogram intercepted, there will be, for the enemy, a certain set of messages with certain probabilities, that this cryptogram could represent. As *N* increases the field usually narrows down until eventually there is a unique 'solution' to the cryptogram; one message with probability essentially unity while all others are practically zero. A quantity *H(N)* is defined, called the equivocation, which measures in a statistical way how near the average cryptogram of *N* letters is to a unique solution; that is, how uncertain the enemy is of the original message after intercepting a cryptogram of *N* letters." [p.659]
>
> "This gives a way of calculating approximately how much intercepted material is required to obtain a solution to the secrecy system. It appears .
> .
> . that with ordinary languages and the usual types of ciphers (not codes) this 'unicity distance' is approximately *H(K)/D*. Here *H(K)* is a number measuring the 'size' of the key space. If all keys are *a priori* equally likely, *H(K)* is the logarithm of the number of possible keys. *D* is the redundancy of the language
> . . . ." "In simple substitution with a random key *H(K)* is $\log_{10}$
> 26! or about 20 and *D* (in decimal digits per letter) is about .7 for English. Thus unicity occurs at about 30 letters." [p.660]

Shannon, C. 1949. Communication Theory of Secrecy Systems. *Bell System Technical Journal.* 28: 656-715.

Uniform Distribution

A probability distribution in which each possible value is equally likely. Also a "flat" or "even" distribution.

A uniform distribution is the most important distribution in cryptography. For example, a cryptographer strives to make every possible plaintext an equally likely interpretation of any ciphertext (see ideal secrecy). A cryptographer also strives to make every possible key equally likely, given any amount of known plaintext.

On the other hand, a uniform distribution with respect to one quality is not necessarily uniform with respect to another. For example, while keyed shuffling can provably produce any possible permutation with equal probability (a uniform distribution of different tables), those tables will have a Boolean function nonlinearity distribution which is decidedly not uniform. And we might well expect a *different* non-uniform distribution for every different quality we measure.

Variable Size Block Cipher

The ciphering concept described in U.S. Patent 5,727,062 (see the VSBC articles on the Ciphers By Ritter page).

A block cipher which supports ciphering in blocks of dynamically variable size. The block size may vary only in steps of some element size (for example, a byte), but blocks could be arbitrarily large.

Three characteristics distinguish a true variable size block cipher from designs which are merely imprecise about the size of block or element they support or the degree to which they support overall diffusion:

1. A variable size block cipher is indefinitely extensible and has no theoretical block size limitation;
2. A variable size block cipher can approach overall diffusion, such that each bit in the output block is a function of every bit in the input block; and
3. A true variable size block cipher does not require additional steps (rounds) or layers to approach overall diffusion as the block size is expanded.

Also see Dynamic Substitution Combiner and Balanced Block Mixing.

Voltage

The measure of electron "potential" in volts. Voltage is analogous to water *pressure*, as opposed to *flow* or current.

Walsh Functions

Walsh Functions are essentially the affine Boolean functions, although they are often represented with values {+1,-1). There are three different canonical orderings for these functions. The worth of these functions largely rests on their being a complete set of orthogonal functions. This allows any function to be represented as a correlation to each of the Walsh functions. This is a transform into an alternate basis which may be more useful for analysis or construction.

Also see: Fast Walsh-Hadamard Transform.

Weight

The weight of Boolean Function *f* is the number of 1's in the truth table of *f*.

Whitening

An overly-cute description of making a signal or data more like white noise, with an equal amount of energy in each frequency. To make data more random-like.

White Noise

A random-like signal with a flat frequency spectrum, in which each frequency has the same magnitude. As opposed to pink noise, in which the frequency spectrum drops off with frequency. White noise is analogous to white light, which contains every possible color.

White noise is normally described as a relative power density in volts squared per hertz. White noise power varies directly with bandwidth, so white noise would have twice as much power in the next higher octave as in the current one. The introduction of a white noise audio signal can destroy high-frequency loudspeakers.

Wire

A thin, long conductor, often considered "ideally conductive" compared to other parts of a circuit.

XOR

Exclusive-OR. A Boolean logic function which is also mod 2 addition.

Terry Ritter, his current address, and his top page.

My Contributions

Terry Ritter

After a decade of full-time work in cryptography, I am sometimes challenged to summarize my work in a form "for mere mortals." Basically I work on automated secret-key ciphers. But the state of the art has come a long, long way from its origin in classic or hand ciphering. First I present the general state of the art, and then I describe my work beyond that.

Modern Secret-Key Ciphering

Modern automatic ciphering benefits greatly both from the availability of computing machinery, and the resulting growth of information-processing knowledge. Thus:

- We have a machine to handle huge numbers of operations quickly and precisely, and
- We have new understandings and new computer techniques.

Consequently, in addition to generalizing the older concepts like:

- Using a 256-element character set (instead of 26)

and eliminating simplifications and their related weaknesses, such as:

- Selecting one of many tables at pseudorandom (instead of in rotating sequence), and
- Transposing *bits* (as opposed to characters or bytes)

we also can:

- Hash arbitrary keys to fixed-size results,
- Build serious pseudorandom sequences (RNG's),
- Use an RNG to key Simple Substitution, and
- Use an RNG to produce message keys and authentication values.

These are things unavailable to classic ciphers, and the result can seem daunting compared to classic cryptograms. But this is the well-known state of the art.

My Supposed Advances

I believe I have made three particular advances beyond the state of the art in a decade of full-time cryptography. Hopefully this will not be completely boring:

1. First is what I call **Dynamic Substitution**. This is basically the use of a modified form of Simple Substitution to replace the additive mixing function in a stream cipher. Each current message character is substituted into ciphertext, then the just-used substitution element is exchanged with some other as selected by a keyed pseudorandom sequence. In this way, the more often a particular transformation is used, the more often it changes, and this destroys usage frequency statistics. The result is a nonlinear combiner with state, and so can be used in a sequence of combinings, or in a selection among different combiners, neither of which make sense with exclusive-OR.
2. Next is what I call **Balanced Block Mixing**, which is useful in constructing block ciphers. The whole point of a block cipher is to produce the same effect as a Simple Substitution with an alphabet size far, far too large for table realization. So the issue becomes one of constructing wide block ciphers (simulated huge tables) from small components.

   A Balanced Block Mixer (BBM) consists of two orthogonal Latin squares; these may be explicit tables or simple computations. The basic BBM is a two-input, two-output function which may be keyed, and is reversible. If we put two small blocks in, we get two small blocks out, but the important point is that each output block depends upon *both* of the input blocks. So if we perform these BBM operations in FFT-like patterns, we can mix a lot of small blocks such that *each* result depends upon *every* input, which is what we need for a block cipher. This allows us to use small keyed invertible substitutions as the confusion basis for a large block cipher.

   The BBM mixing concept is especially interesting for hardware realization, where each BBM across the block can be implemented in separate hardware so all may function in parallel. This means that we can achieve a throughput which is constant *per block*, so the larger the block, the more data ciphered in the same amount of time. (Latency does increase, but the ciphering rate is fixed per block by the technology, which also limits the maximum on-chip block size.)
3. Last is what I call **Variable Size Block Ciphers**. These have the usual block cipher characteristics, but can cipher blocks of virtually arbitrary size (beyond some minimum) *to the byte*, as dynamically selected at ciphering time. This occurs in the context of a "constant depth" process, which needs no additional computation per byte, even with huge blocks, so there is no motive to cipher data in small blocks.
   - VSBC's can support huge blocks and so allow a return to ECB-mode (Electronic Code Book) operation, which is frowned-upon with the government standard DES.
   - VSBC's can support ciphering already-defined fields in existing systems without the need for added Initial Value (IV) storage. While an IV is not necessarily difficult to make or use, it does expand the amount of store needed. This makes it difficult to cipher wholly within existing data structures.
   - VSBC's also support the ability to hide both the start and end positions of each block. (We send some random data at the start, and again at the end, and use blocks of keyed pseudorandom size in between.) This ability to hide the extent of a block directly confronts one of the things we normally assume an Opponent will have in any attack on a block cipher.
   - VSBC's make it easy to expand the block to contain control fields, for dynamic keying, authentication, or both.

I have also found a couple of interesting simple combinatoric relationships:

1. One is related to the Birthday Paradox. If we are willing to admit a new definition for what I call "augmented" repetitions, very simple and reversible formulas predict the number of augmented doubles, triples, etc. for random sampling from a given population. The point of this, of course, is to be able to estimate a huge population size fairly accurately from trials with on the order of square-root-of-the-population samples. This is the reason we need exact reversible formulas. Such measurement can help to certify hardware really-random number generators (RNG's), and perhaps also realistically estimate the number of unique keys in small versions of block ciphers.
2. More recently, I have identified the combinatoric model for runs-up and runs-down RNG tests, and so eliminate a statistical bias which one gets using the approximation in Knuth II.

All this stuff is discussed in horrendous detail on my pages. I try to update and improve my pages monthly, but there is very little glitter there.

Terry Ritter

Terry Ritter, his current address, and his top page.

Last updated: 1998-02-24

Random Noise Sources

PRELIMINARY Noise Source Experiments

A *Ciphers By Ritter* Page

Terry Ritter

1999-03-30

It has been long assumed that physically generated random noise can yield cryptographically unpredictable random numbers. These **preliminary** designs are the first step in a several-step process of producing cryptographic random numbers.

Any electronics experimenter can build a noise source. Unfortunately, building a *good* noise source is harder than it looks. Correctly using such a source is harder still.

Post-Processing

In these designs, I assume that subsequent post-processing is not optional, but required. The production of analog noise is just the first step in producing cryptographic values. The advantage in raw noise is the ability to see the quality of the generator separate from the cryptographic output. In most cases, post-processing will hide major defects in the random device. Without post-processing, we should see the generator values fit a distribution corresponding to the noise source itself, so we can have confidence that we are seeing real random noise. Only *then* can we in confidence process the data to accumulate "entropy," remove statistical "bias," and return "bits."

Strategies

I have constructed a sequence of analog noise generators for analysis. My first approach was a zener noise circuit, using a simulated zener IC to support low-voltage operation. The second was just a consumer FM radio, with noise from semiconductor RF amplification and mixing taken directly from headphone output. Subsequent realizations include a pseudonoise generator (to calibrate the noise card), a narrowband IF strip, and a voltage-mode generator with improved white output.

For those generators intended for actual random number usage, the generators are battery powered, so there is no AC "hum" or switching noise which might be mistaken for quantum noise. They are completely enclosed in their own ferrous and electrically-conductive "tea tin," so there is no magnetic, electrostatic, or electromagnetic field sensitivity to worry about. (If we tune the FM receiver to a local station, we can put the top on the can and listen through the can as the signal drops into noise.) The only hole in each enclosure is the shielded RCA jack used to deliver the analog noise.

Target: Sound Card

The intent of these generators is to deliver analog noise to be digitized by an ordinary sound card. Since we have to amplify our low-level noise, it seems like we could reduce our effort by using the sound card microphone input. Unfortunately, the usual SB16 sound card (there are a variety of *different* SB16 models) needs about 100 mv from the mike, at low impedance. This is a *huge* signal for a microphone; it implies an internal amplifier. The microphone input on my SB16 has a measured input impedance of about 2k to ground, which is unexpectedly low.

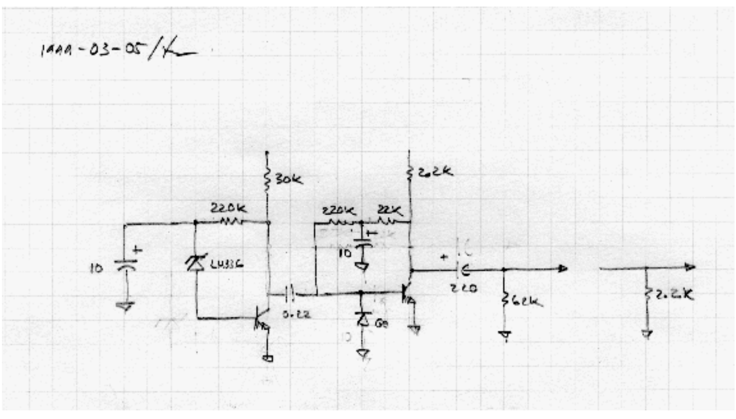
The SB16 microphone input connection is a stereo miniature jack, but only the "tip" contact is used for signal; the "ring" contact is used to supply power. The power is apparently used by FET's in the internal amplifier of the electret type condenser microphones which are now common. This internal amplification is presumably how we can expect such a large signal from a microphone.

Some web searching turns up a variety of comments by people who got voice-recognition software and tried to use, for example, a dynamic mike in their sound card. While I have been unable to find input level and connection specifications from the card manufacturer, there are some comments on sound card microphone inputs in the Sound Blaster Microphone Preamplifier page.

We will just be supplying noise signal to the sound card, and not using the voltage from the card, thus avoiding any possibility that the power line might affect the quality of the delivered noise.

The "First Cut" Zener Noise Design

My first approach was to produce zener noise, which has been used many times. But in published circuits, this always had the problem that higher voltages were required than we might prefer. The voltage problem is attacked here by using an IC "zener" device, which has a low zener voltage. Also, some past designs have used digital circuits in so-called linear modes, again to support low-voltage operation. The intent here is to pretty much use the components the way they were intended to be used.



This circuit functions below 3 volts! Part values are generally non-critical, except for the LM336, which is fairly common.

The IC Zener LM336

A zener diode is a semiconductor junction operated in reverse bias breakdown, and built to take the power dissipation of such operation. Good low-voltage zeners are not available, which is why we have a simulated zener in an IC.

The low zener voltage supports long battery life in operation. Moreover, the spec sheet clearly lays out the noise of the device. We can hope to find that very same frequency response in our results, which would be a strong indication that we were measuring that device, as opposed to some other unknown signal.

The 2N2222 Transistor

The first transistor is used to convert current noise from the zener into amplified current noise in the collector, which is seen as voltage noise across the collector resistor. Since bipolars amplify current, sensing current noise seems to be more appropriate than sensing voltage noise.

Every bipolar transistor must be biased properly: here we want the collector to rest at some reasonable middle level so that noise peaks are not cut off in either direction. We can do that by using the zener itself, which apparently forces about 3.1 v on the collector. (We operate the zener at very low currents, however, so it is out of regulation and has a lower voltage.) The collector noise signal is filtered out by the capacitor, giving an average level to the zener. The capacitor also provides a current "ground" for the noise signal produced by the zener and flowing through the transistor base.

The second transistor amplifies the signal to a level usable by the sound card microphone input. It has two different bias systems: first, filtered feedback bias which does not affect the signal gain, and second, a germanium diode on the transistor base. The diode clamps signals below ground to about -0.3 v, but more importantly charges the input capacitor from negative peaks, which thus provides additional bias to avoid such peaks in the future.

The 2N2222 is my favorite general-purpose experimental transistor: it handles reasonable voltage and power, and has good gain with reasonable linearity. But there is nothing magic about this device, and pretty much any small-signal NPN will do here. As far as I know, the 2N2222 is still in production after almost 30 years, and it fits our requirements. Modern transistors are not better if we do not use whatever advantages they may have: We have no need for high voltage, power, frequency or gain.

AirBoard Construction

To see how a circuit performs we must construct it. The hard way would be to lay out a printed-circuit board, drill it and populate it. This is a long delay, and a lot of work for something which may be wildly changed.

Another way to construct the circuit would be to insert the parts through perforated circuit board. This is better, but is still too much overhead for new designs where every part may change (and thus must be removed and replaced) during development.

So instead of a first "breadboard," I construct an "AirBoard": I usually start with power and ground bus wires, then tack-solder parts to those, and to each other. If we leave these leads long we can connect most parts directly, in a three-dimensional volume. More complex circuits will benefit from insulating the leads and partitioning into smaller developments.

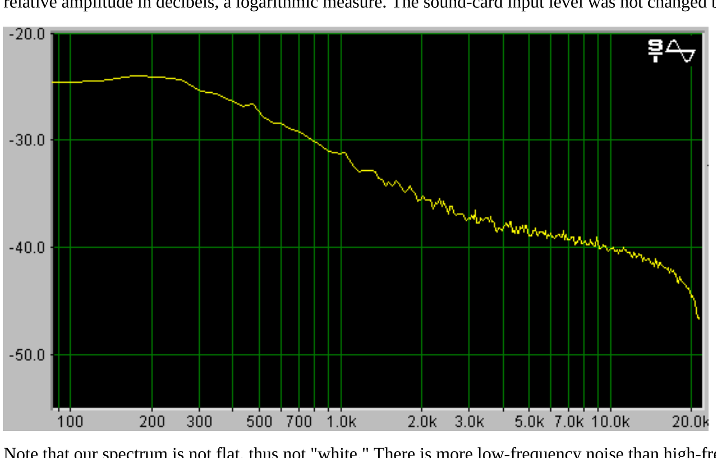
Zener First Cut Measured Performance

We design a circuit to generate noise, implement the circuit, then record that noise using a sound card. All of the recordings here are CD-quality monaural: 16-bit samples at 44.1 KHz. These are very large files for short sounds.

- Zener noise, 1.7 sec, 150K
- Zener noise, 9.0 sec, 794K

Once we have recorded the noise, we can inspect the spectral qualities by using a Fast Fourier Transform (FFT). Here we use the Spectra PLUS FFT display by Sound Technology Inc. The FFT display gives us the ability to see what we have, and the effects of our changes. Spectra PLUS is a fabulous tool, and a spectral display is necessary for noise hardware development. But Spectra PLUS is also very expensive, with irritating option pricing and copy-protection. It is to be hoped that marketplace competition will solve these problems.

All of the FFT results shown here are the average of 100 different 1024-point FFT's using a Hamming window. This is random noise, and any one FFT computation can look peculiar; only the average of many such computations gives the (relatively) smooth curve we might at first expect. The numbers across the bottom (100 to 20.0k) represent frequency in Hertz; the numbers on the left represent relative amplitude in decibels, a logarithmic measure. The sound-card input level was not changed between any of these FFT's, so these relative levels have the same meaning in each graph.



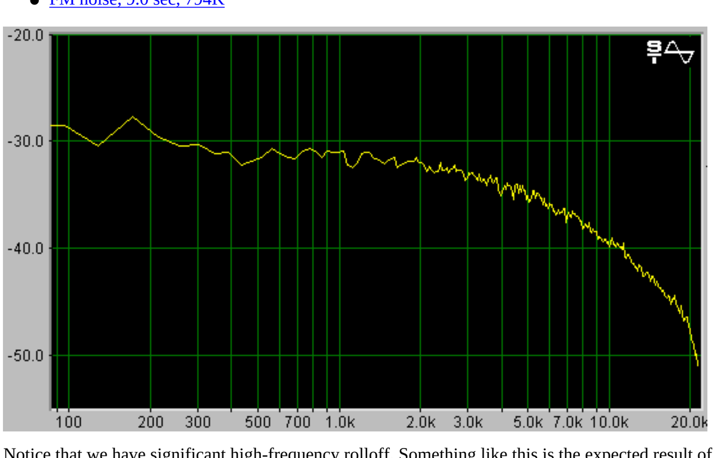
Note that our spectrum is not flat, thus not "white." There is more low-frequency noise than high-frequency noise, so our noise is "pink." Technically, for ideal pink noise, the amplitude should decrease by 3 dB per octave, which is about what we see here. Since a single RC filter will roll off at 6 dB per octave, it is not possible to use a simple filter on white noise to get pink noise.

The Consumer FM Noise Approach

A battery-powered consumer FM radio is placed in a metallic can as a shield. The radio is placed in monaural mode, and the output taken across one of the earphones. An appropriate level for the sound card is slightly audible a foot or two away.

Consumer FM Measured Performance

- FM noise, 1.7 sec, 150K
- FM noise, 9.0 sec, 794K



Notice that we have significant high-frequency rolloff. Something like this is the expected result of 75 usec "de-emphasis" time-constant in the broadcast FM process. We do seem to approach the expected -6 dB / octave slope at the high frequencies.

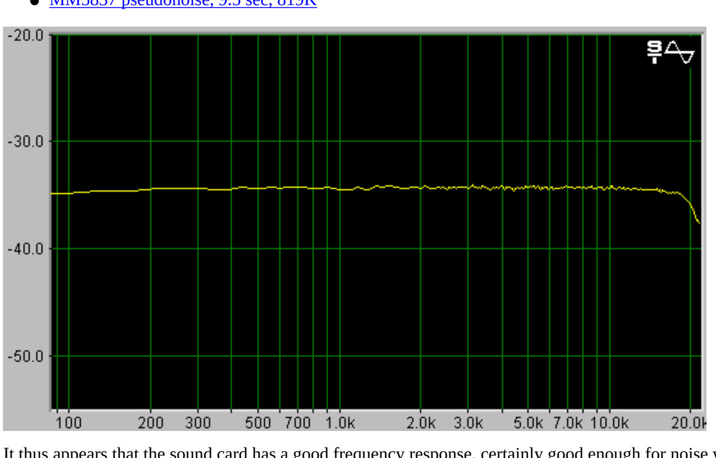
(Noise in FM reception tends to be "white," which has the same energy in each Hz of bandwidth. This means that most energy is in the high frequencies, which is annoying to humans. We can minimize this by *pre*-emphasizing the transmitted sound, so that the audio highs are magnified or emphasized in the transmitted signal. Then we *de*-emphasize highs in the receiver, which cuts the noise, and returns the audio system response to approximately flat.)

A Pseudonoise Spectrum

As a check on the capabilities of the sound card, I implemented a noise generator based on the old MM5837 Digital Noise Generator IC. I suspect that we could do better with a CMOS LFSR, but this was simpler.

Pseudonoise Measured Performance

- MM5837 pseudonoise, 1.7 sec, 148K
- MM5837 pseudonoise, 9.5 sec, 819K



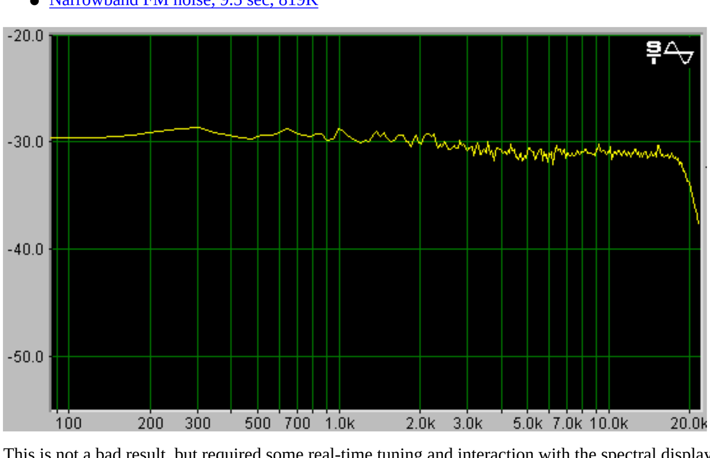
It thus appears that the sound card has a good frequency response, certainly good enough for noise work. It seems likely that the dip at the high end is the start of the necessary low-pass filter to prevent aliasing effects from the 44.1 kHz sampling rate. (Suppose we have a signal of 28 kHz and sample at 44.1 kHz; we can expect to see a spurious 16 kHz component which does not in fact exist. This is a result of sampling a frequency above the Nyquist limit, which is half the sampling rate.)

Noise from Narrowband 10.7 MHz FM IF Strip

At one time it made sense to have a separate IF and audio section into which various front-end RF down converters could be connected. In such a section we have some tuned circuits and a mixer which down converts the 10.7 MHz IF to 455 KHz. The signal is amplified and limited at 455 KHz, and (apparently) slope-detected. The de-emphasis components were opened and audio taken before the on-board amp. The intent was to view so-called "FM noise."

Narrowband FM Measured Performance

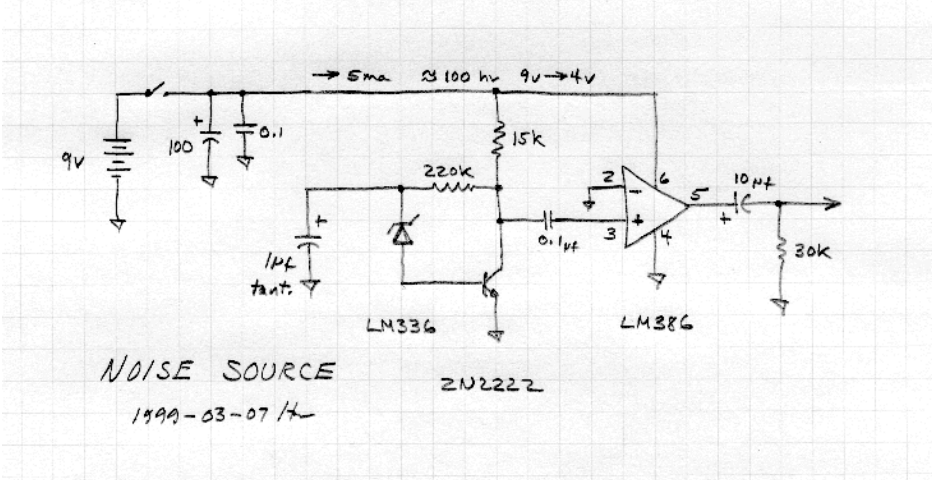
- Narrowband FM noise, 1.7 sec, 146K
- Narrowband FM noise, 9.5 sec, 819K



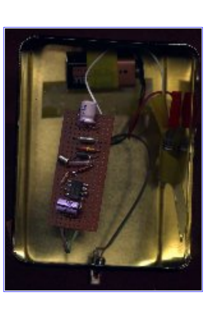
This is not a bad result, but required some real-time tuning and interaction with the spectral display to achieve. It is also a very technical solution, with lots of amplification and tuned circuits. The noise source itself was localized (by opening the circuit at various stages) to the first stage, a mixer transistor. Presumably, we see amplified transistor noise from 10.7 MHz. (It would be interesting to see how a PLL detector would respond; if we could eliminate the de-emphasis, we might get an FM noise-source without coils.)

The "Second Cut" Zener Noise Design

To improve the overall linearity, an LM386 audio amp chip replaces the output transistor.



The LM386 Audio Amp Chip

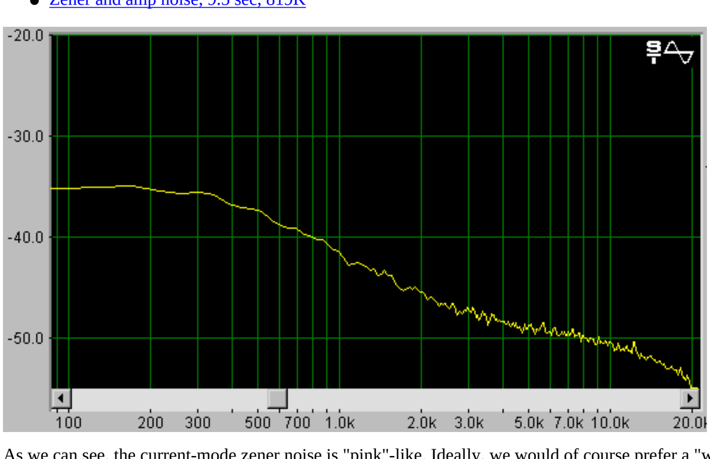


The LM386 is not a high-power amp, but we do not need power. It is simple, but more importantly, it operates over the wide voltage range we expect to encounter in battery operation. The first cut worked down below 3 volts; this new version is limited by the output amp to about 3.5 volts (4 volts guaranteed). This is still very good.

The rated life of a 9-volt battery generally assumes that the output will decline to 4 volts. If we can use this low voltage, we can use the full energy resources of the battery, which is about 580 ma/h for an alkaline version. Since the circuit works to 4 volts, and uses about 5 ma, we expect about 100 hours of operational life per battery. That is a lot of bits.

Zener and Amp Measured Performance

- Zener and amp noise, 1.7 sec, 148K
- Zener and amp noise, 9.5 sec, 819K

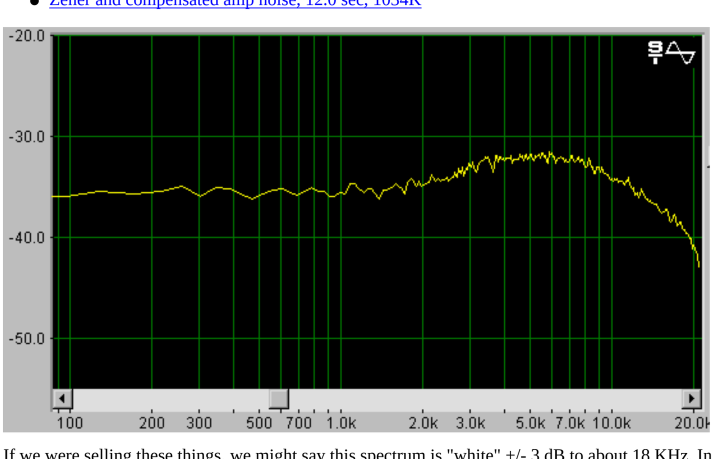


As we can see, the current-mode zener noise is "pink"-like. Ideally, we would of course prefer a "white" response, although that may not matter after processing.

Not shown in the above diagram are "gain pins" 1 and 8 on the LM386 chip. By connecting a capacitor between those pins, we can increase the gain by another 20 dB. By selecting the value of that capacitor, we can select the frequency at which the gain increase takes hold. By adding a 0.22 uF cap across the gain pins, we can compensate the response, to an extent.

Zener and Compensated Amp Measured Performance

- Zener and compensated amp noise, 1.7 sec, 148K
- Zener and compensated amp noise, 12.0 sec, 1034K

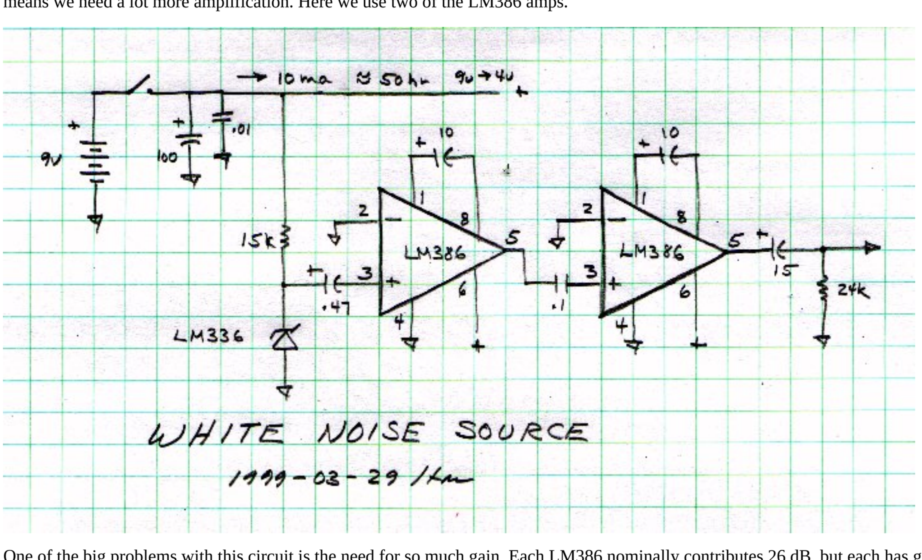


If we were selling these things, we might say this spectrum is "white" +/- 3 dB to about 18 KHz. In fact, we are "equalizing" the very pink signal spectrum, and there is just so much we can do. Eventually the 20 dB additional gain of the LM386 has been used up, while the pink roll off continues. Here we over-compensate a little early to stave off the eventual decline.

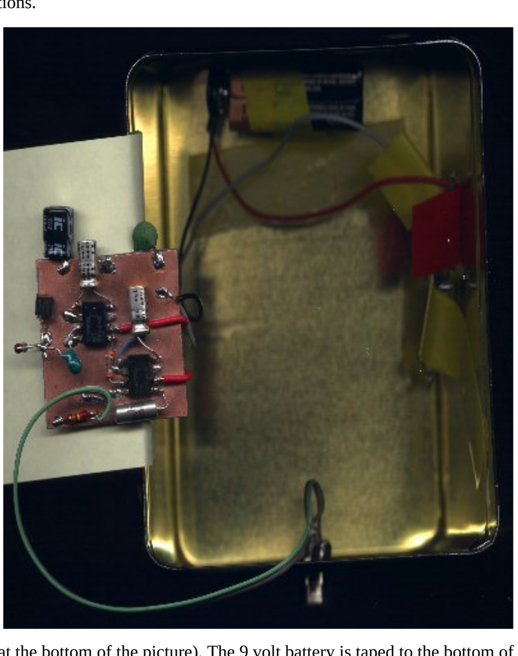
Again, we have no reason to believe that we need a perfect "white" source to generate random data, provided we use substantial post-processing. Since we assume such processing, this could be a very acceptable noise source in actual use.

Voltage Mode Zener Noise Design

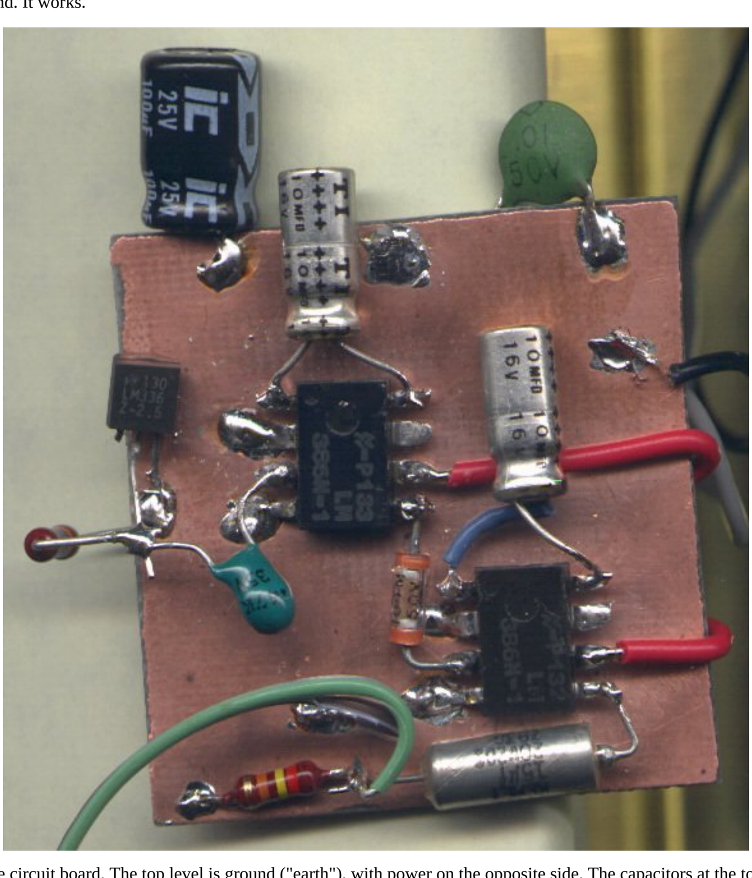
Previously, we have amplified zener noise current, which seems to produce a generally pink spectrum. The alternate approach would be to amplify zener noise voltage. This is very tiny, however, which means we need a lot more amplification. Here we use two of the LM386 amps.



One of the big problems with this circuit is the need for so much gain. Each LM386 nominally contributes 26 dB, but each has gain pins which make it 46 dB, each. This is 92 dB total gain, which can easily get out of hand, producing various oscillations and interactions.



Here we have the ordinary tea-tin, with an RCA jack in the end (at the bottom of the picture). The 9 volt battery is taped to the bottom of the can, and a mercury switch is taped at an angle on one side. We turn the generator "on" by tipping it up on end. It works.



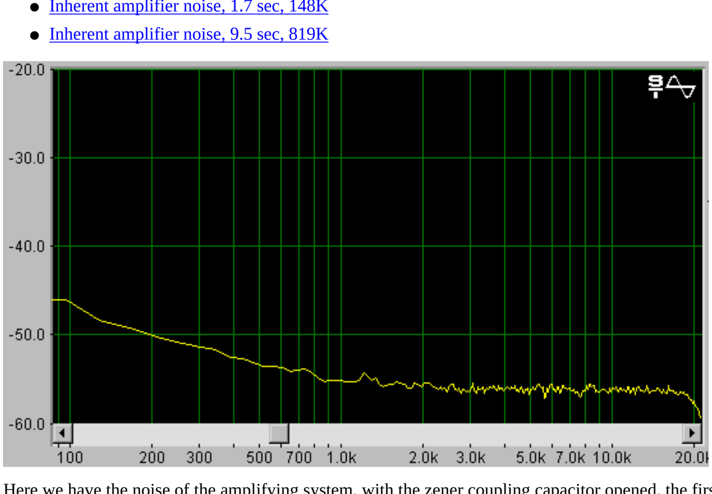
This version is built on a bit of double-sided bare circuit board. The top level is ground ("earth"), with power on the opposite side. The capacitors at the top filter power; the red wires at the right deliver power from the other side of the board. (Had there been oscillation trouble, I would have drilled through by the power pins and supplied power with an 1/8 inch lead from power-plane, which is what I do for digital circuits.)

Depending on our video display settings, we should be able to see most component values; the obvious exception being the vertical 15K resistor from the power side to the zener IC. The chips have their leads cut short with the ground pin and one input pin soldered to the ground plane.

The intent of placement was to try to distance low-level input signals from the resulting high-level output. A slightly larger board might have helped keep the output capacitor completely on the high-level end. And I could have put the zener chip closer to the top.

Voltage Mode Performance: Input Open, No Gain Caps

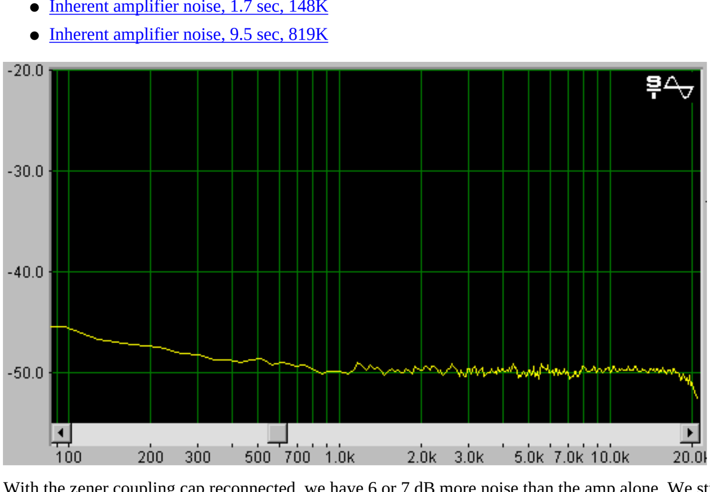
- Inherent amplifier noise, 1.7 sec, 148K
- Inherent amplifier noise, 9.5 sec, 819K



Here we have the noise of the amplifying system, with the zener coupling capacitor opened, the first amp thus having only the internal 50K input resistor. To the left we have the typical 1/f semiconductor noise. Note that the amplitude range has been extended to 60 dB for this spectrum only; without that, most of the graph was below the window.

Voltage Mode Performance: Zener Noise, No Gain Caps

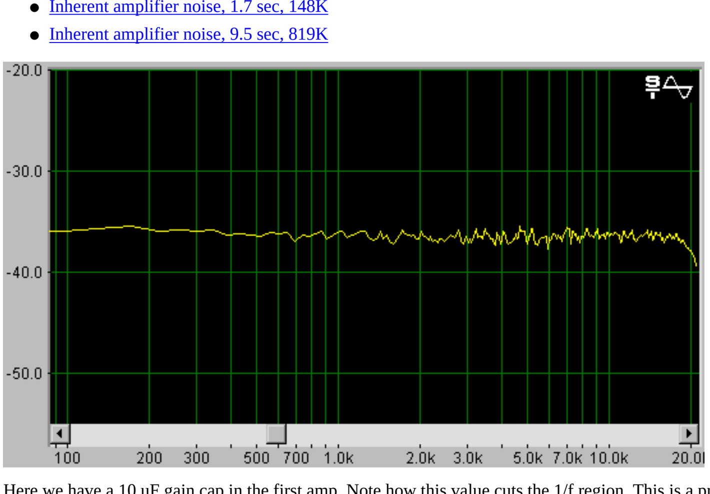
- Inherent amplifier noise, 1.7 sec, 148K
- Inherent amplifier noise, 9.5 sec, 819K



With the zener coupling cap reconnected, we have 6 or 7 dB more noise than the amp alone. We still have that 1/f region, however, and the amplitude is pretty low.

Voltage Mode Performance: Zener Noise, One Gain Cap

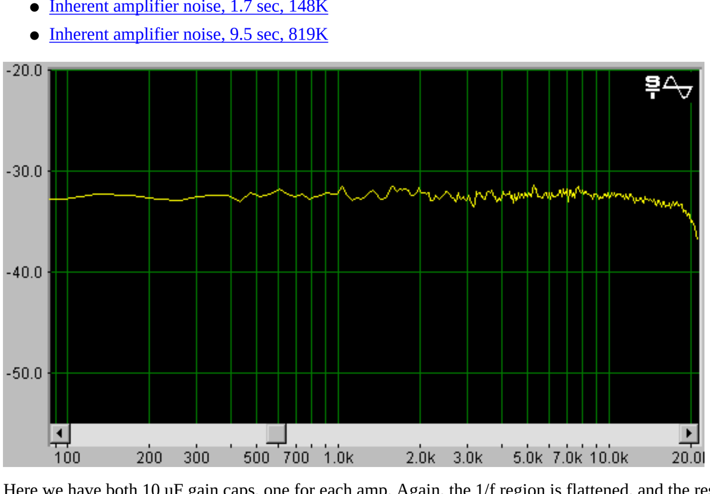
- Inherent amplifier noise, 1.7 sec, 148K
- Inherent amplifier noise, 9.5 sec, 819K



Here we have a 10 uF gain cap in the first amp. Note how this value cuts the 1/f region. This is a pretty good spectrum, maybe +/- 1 dB to 20 KHz. The amplitude is a little low, though.

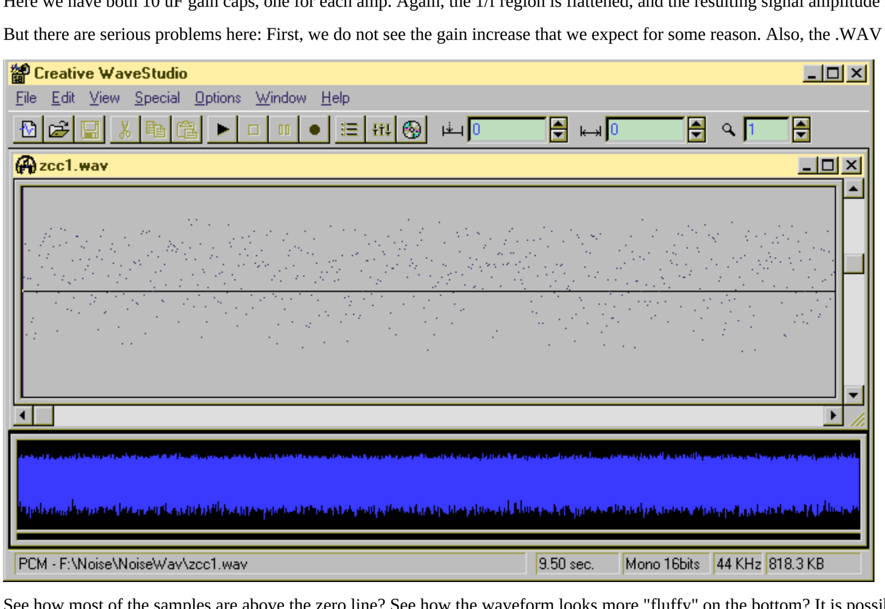
Voltage Mode Performance: Zener Noise, Both Gain Caps

- Inherent amplifier noise, 1.7 sec, 148K
- Inherent amplifier noise, 9.5 sec, 819K



Here we have both 10 uF gain caps, one for each amp. Again, the 1/f region is flattened, and the resulting signal amplitude is about what we had with the earlier transistor and single-amp system.

But there are serious problems here: First, we do not see the gain increase that we expect for some reason. Also, the .WAV file looks biased.

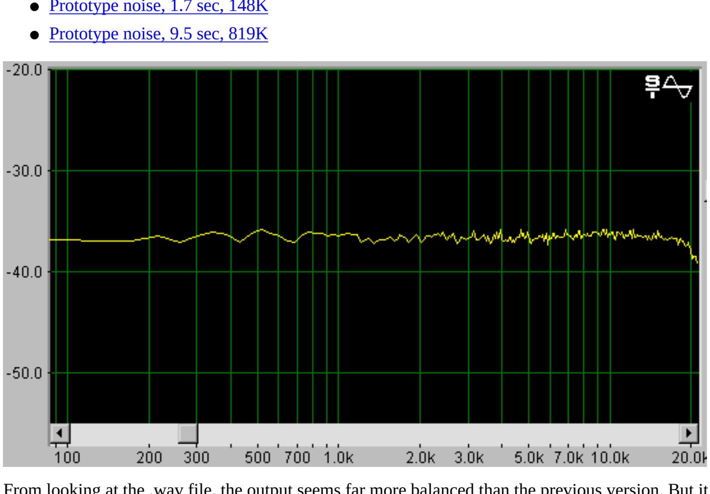


See how most of the samples are above the zero line? See how the waveform looks more "fluffy" on the bottom? It is possible that we are somehow clipping some of the signal. So even though the spectrum is about as good is we can expect from a real device, we are probably better off leaving the gain pins open on the second amp.

In most of these designs, the signal is capacitively coupled to the (+) amplifier input. Ideally, we would similarly couple the (-) input to ground, which would reduce output offset. But in the prototype with an 8.7 volt supply, we have 4.6v and 4.4v on the two amplifier outputs, and these are very reasonable values.

Current Prototype Performance (Zener, 2 Amps, 1 Gain Cap)

- Prototype noise, 1.7 sec, 148K
- Prototype noise, 9.5 sec, 819K



From looking at the .wav file, the output seems far more balanced than the previous version. But it will take some detailed analysis before we know just what we have.

Last updated: 1999-03-30

sci.crypt #47999
Path: news.io.com!io.com!not-for-mail
From: ritter@io.com (ritter)
Newsgroups: sci.crypt

Subject: Variable Size Block Cipher Designs
Date: 11 Feb 1996 16:23:09 -0600
Organization: Illuminati Online
Lines: 294
Message-ID: <4flq8d$gf4@pentagon.io.com>
NNTP-Posting-Host: pentagon.io.com

From my previous work on Variable Size Block Ciphers (VSBC) I have realized a family of five very interesting cipher architectures.

This is another set of ciphers you won't find in the crypto books.

These working prototypes are intended to explore a range of "Variable Size Block Cipher" designs, and provide some crude performance comparisons. This is patent-pending work.

VARIABLE SIZE BLOCK CIPHERS

The usual block cipher has a small fixed width (often 64 bits). But if plaintext really does contain uniqueness at a rate of about one bit per character, the usual block cipher covers only about eight bits of uniqueness, a searchable quantity. (This sort of attack, of course, would require a deep knowledge of language structure, but doing "Triple" anything would not affect it.) By increasing block size dramatically, we can have a cipher which is not vulnerable to this sort of attack.

The usual stream cipher can cipher sections of arbitrary size, but can only propagate uniqueness to *later* ciphertext. In contrast, a block cipher -- including all of these VSBC designs -- will propagate plaintext uniqueness to *all* bits of ciphertext in the same block.

The VSBC concept is a new paradigm, and can be used in ways far different than fixed size block ciphers. For example, in a VSBC there is no need to use fixed-size blocks: A message can be ciphered in blocks of *pseudo-random* size. This means that an Opponent cannot even know what data belongs to a single block.

THE PROTOTYPE PROGRAM

The VSBC prototype program uses any of these VSBC designs and ciphers data in random-size blocks of from 10 to 4105 bytes (the average data block is about 2 KB). Random length "pre" and "post" blocks (an average of 1/2 K each) which do not contain message data are used to hide the first and last data block boundaries.

```
                A VSBC CIPHERTEXT FILE
   +-----+----------------------+----/ /----+-----+
   | Pre |      Data Block      |           | Post|
   +-----+----------------------+----/ /----+-----+
```

The size of each block is generated by the crypto RNG operating under some key. (In an actual cipher product, the User Key would decipher an Alias file, in which a public alias would be matched to recover a corresponding Alias Key. That Alias Key would be used to decipher only the Message Key from a particular message from a particular correspondent. The Message Key -- which might be very long -- would only then initialize the RNG for deciphering data. This same process is actually implemented in my Penknife and Cloak2 commercial ciphers.)

A typical VSBC file may not even *have* two blocks of the same size, and there would seem to be no way to determine the start or end of any block. This would seem to be a significant strength advantage, *beyond* the strength of the cipher design, flowing directly from the new concept of a dynamically-variable block size.

```
              A VSBC CIPHERTEXT DATA BLOCK
     +---------------+-------+----------/ /---------+
     |      VCF      |  Len  |                      |
     +---------------+-------+----------/ /---------+

     |<--- 4 By ---->| 2 By  |<--- 10..4105 By ---->|
```

Four pseudo-random validation / error-check bytes are added to each block when enciphering. When deciphering, the recovered Validation Check Field (VCF) is compared to the pseudo-random value from the crypto RNG, and any error indicated. This avoids an additional pass over the data to compute a CRC for error-detection, creates a validation field which is not easily spoofed, and provides a form of dynamic block-by-block keying. Clearly, the VCF could be of any desired size.

Two bytes are used to specify the amount of data in the *next* block. This value is needed to specify the length of the last data block, which is usually short by some unknown amount.

Typically six "iv" or Initialization Value bytes (again developed from the crypto RNG) are used by some of these VSBC designs, and the resulting carry values are used as iv's for the next block. This provides an automatic form of block chaining and randomization for the leftmost and rightmost column inputs. Those designs which use substitution tables step through an array of 64 such tables. The starting table value is also initialized from the crypto RNG and the "next up" table is also chained from block to block.

THE DESIGNS

Here I show the structure for typically three adjacent bytes, but each of these designs is used in a file cipher which dynamically selects block size to the byte with an average size of about 2 KB.

These VSBC designs have a very similar structure for both enciphering and deciphering. Data flows down for enciphering, and up for deciphering. The tables (or Latin square rows) used for deciphering are the inverse of those used for enciphering. In the SBBM and OLs designs, the diffusion direction also changes.

In the two designs which use substitution tables, the tables are numbered to remind us that each is generally different. Nor is the table for any particular position fixed: Tables are used as needed from an array of 64 such tables. Each table is separately shuffled by the cryptographic RNG and each table represents a potential 1648-bit key. Normally, the keyspace of this sort of cipher is limited by the size of the RNG used to key the tables or squares, and it is easy to have an efficient true keyspace of many thousands of bits.

In marked contrast to other cipher designs, additional confusion and diffusion layers are easily and modularly added to produce a cipher of any desired working strength.

Performance measurements occurred for RAM-drive ciphering of a 750K file on a P90 under DOS 6.22, with single-pass shuffles.

SubX (init = 15 ms, ciphering = 461 KB/sec)

```
              INPUT or OUTPUT
        d[0]        d[1]        d[2] . . .
         |           |           |
 iv0 -> XOR  +----> XOR  +----> XOR
         |   |       |   |       |
        S00  |      S10  |      S20
         *-- | -+    *-- | -+    *--> c1
 iv1 -> XOR  |  +-> XOR  |  +-> XOR
         *---*       *---*       *--> c0
 iv2 -> XOR  |  +-> XOR  |  +-> XOR
         *-- | -+    *-- | -+    *--> c2
        S01  |      S11  |      S21
         |   |       |   |       |
 iv0 -> XOR  +----> XOR  +----> XOR
         |           |           |
        XOR <----+  XOR <----+  XOR <- iv4
         |       |   |       |   |
        S03      |  S13      |  S13
   c5 <--*    +- | --*    +- | --*
        XOR <-+  |  XOR <-+  |  XOR <- iv5
   c4 <--*       *---*       *---*
        XOR <-+  |  XOR <-+  |  XOR <- iv6
   c6 <--*    +- | --*    +- | --*
        S04      |  S14      |  S14
         |       |   |       |   |
        XOR <----+  XOR <----+  XOR <- iv4
         |           |           |
        d[0]        d[1]        d[2] . . .
               OUTPUT or INPUT
```

The SubX VSBC design uses exclusive-OR combining in diffusion layers, and byte-wide 256-entry substitution tables in confusion layers. The particular table used in each position is the "next" table in sequence from an array of keyed tables. (Even if two blocks do end up having the same size, they probably will not have the same tables in the same positions.)

The SubX design uses the most primitive components, and so is graphically more complex than the other designs.

Ls4 (init = 55 ms, ciphering = 236 KB/sec)

```
        d[0]        d[1]        d[2] . . .
         |           |           |
 iv0 -> Lsc  +----> Lsc  +----> Lsc
         *-- | -+    *-- | -+    *--> c1
 iv1 -> Lsc  |  +-> Lsc  |  +-> Lsc
         *---*       *---*       *--> c0
 iv2 -> Lsc  |  +-> Lsc  |  +-> Lsc
         *-- | -+    *-- | -+    *--> c2
 iv0 -> Lsc  +----> Lsc  +----> Lsc
         |           |           |
        Lsc <----+  Lsc <----+  Lsc <- iv4
   c5 <--*    +- | --*    +- | --*
        Lsc <-+  |  Lsc <-+  |  Lsc <- iv5
   c4 <--*       *---*       *---*
        Lsc <-+  |  Lsc <-+  |  Lsc <- iv6
   c6 <--*    +- | --*    +- | --*
        Lsc <----+  Lsc <----+  Lsc <- iv4
         |           |           |
        d[0]        d[1]        d[2] . . .
```

The Ls4 (four Ls's per data element per diffusion direction) VSBC design uses Latin square combining, which simultaneously provides both diffusion and confusion in the same layer. A single Latin square of order 256 is used, requiring 64K of store and representing a 3296-bit key. (The prototype RNG is a jitterized Additive RNG with 496 bits of initial state.)

The overall structure is quite like the SubX design, but differs in that the "table" (actually, the Ls row) used at each node is here selected by diffusion *data*, instead of some value related to node position. Since The Opponent does not see the diffusion data, it is going to be tough to isolate a particular "table" to be attacked separately.

VSBC LsX (init = 55 ms, ciphering = 372 KB/sec)

```
        d[0]           d[1]           d[2] . . .
         |              |              |
 iv0 -> XOR  +-------> XOR  +-------> XOR
         *-- | ----+    *-- | ----+    *--> c2
 iv1 -> Lsc  |  +----> Lsc  |  +----> Lsc
         *---*  |  |    *---*  |  |    *--> c0
 iv2 -> Lsc  |  |  +-> Lsc  |  |  +-> Lsc
         *-- | -+      *--- | -+       *--> c1
 iv0 -> XOR  +-------> XOR  +-------> XOR
         |              |              |
        XOR <-------+  XOR <-------+  XOR <- iv4
   c6 <--*    +---- | --*    +---- | --*
        Lsc <----+  |  Lsc <----+  |  Lsc <- iv5
   c4 <--*    |  |  *---*    |  |  *---*
        Lsc <-+  |  |  Lsc <-+  |  |  Lsc <- iv6
   c5 <--*       +- | --*       +- | --*
        XOR <-------+  XOR <-------+  XOR <- iv4
         |              |              |
        d[0]           d[1]           d[2] . . .
```

The LsX design uses Latin square layers for both diffusion and confusion, and exclusive-OR combining for other diffusion layers. This design also demonstrates a somewhat different feedback architecture.

SBBM (init = 15 ms, ciphering = 439 KB/sec)

```
   d[0]    d[1]    d[2]    d[3] . . .
    |       |       |       |
   S00     S10     S20     S30
    |       |       |       |
    +----> BBM --> BBM --> BBM -->      -----+
            |       |       |                |
           S01     S11     S21              Sy1
            |       |       |                |
            +----> BBM --> BBM -->      --> BBM -----+
                    |       |                |       |
                   S02     S12              Sx2     Sy2
                    |       |                |       |
            +----- BBM <-- BBM <--      <-- BBM <----+
            |       |       |                |
           S03     S13     S23              Sy3
            |       |       |                |
    +----- BBM <-- BBM <-- BBM <--      <----+
    |       |       |       |
   S04     S14     S24     S34
    |       |       |       |
   d[0]    d[1]    d[2]    d[3] . . .
```

The SBBM VSBC design uses the fast and simple Balanced Block Mixing component in the diffusion layers, and byte-wide substitution tables in the diffusion layers. Again, these tables are used in cyclic sequence from an array of keyed tables, so the same tables may or may not occur in the same positions in some other block.

OLs (init = 76 ms, ciphering = 369 KB/sec)

```
   d[0]    d[1]    d[2]    d[3] . . .
    |       |       |       |
    +----> OLs --> OLs --> OLs -->      -----+
            |       |       |                |
            +----> OLs --> OLs -->      --> OLs -----+
                    |       |                |       |
            +----- OLs <-- OLs <--      <-- OLs <----+
            |       |       |                |
    +----- OLs <-- OLs <-- OLs <--      <----+
    |       |       |       |
   d[0]    d[1]    d[2]    d[3] . . .
```

This OLs VSBC design uses a keyed orthogonal pair of Latin squares which realize a keyed Balanced Block Mixer at each node. The OLs structure fills 128K and also represents 3296 bits of key.

Your Comments Wanted

All comments on these designs, positive or negative, will be appreciated. Serious financial support is needed, but no personal donations, please. Companies interested in these or other future ciphering technologies could do worse than to establish a continuing relationship through contractual development.

Terry Ritter   Ciphers By Ritter   http://www.io.com/~ritter/

Extreme Hardware Speed in Large Block Mixing Ciphers

Ciphering on Chip: How Big is Too Big?

Terry Ritter

The goal of this exercise is to get some idea of the chip area required to encipher a large block with Mixing cipher technology.

Can We Expect to Efficiently Cipher Large Blocks?

An interesting consequence of ciphering a large block is the possibility of extreme hardware speed. Although we expect hardware to be fast, in some cases, throughput can *increase* with block size, which may be unexpected. After all, in a block cipher, it is necessary to in some way "mix" the various parts of the block, and it seems that a larger block *must* need more mixing. It does, but certain mixing constructions lend themselves to parallel hardware which can track block size increases.

Over a period of several years I have developed Balanced Block Mixing (BBM) structures for invertible ciphering which offer arguably "perfect" mixing across wide blocks. Byte-size BBM's can be used in FFT-like patterns to mix huge blocks with complexity n log n.

In a Mixing cipher, each of the BBM operations is independent, and uses and updates two values which are not simultaneously in use by other BBM's. This means that we can realize each BBM in separate hardware, and have them function in parallel. We can push this to unexpected heights.

How Much Chip Do We Need?

To get a general idea about how much of a chip the design would consume, we can look at a few chip examples in current production:

- Static-memory devices typically lag dynamic memory sizing by somewhat more than a generation. Current Motorola SRAM chips include a range of 4mb (4 megabit) designs.
- Individual customizable gates are larger than static RAM cells. As data points, Altera has a 250k gate FPGA, and the average IBM ASIC is said to have been about 260k gates in 1996.

One might thus conclude that "a gate" is as large as 16 static RAM cells, on average, which seems a little high. But the actual value does not matter much here, because the design is heavily dominated by RAM anyway.

A Mixing Cipher

The mixing cipher described here has a fencing layer (a row of substitution tables), Balanced Block Mixing, fencing, mixing and fencing. This is a conceptually simple cipher.

In a mixing cipher, the tables constitute primary keying and are set up by an external subsystem prior to ciphering.

A Balanced Block Mixer

The Mixing cipher uses Balanced Block Mixer or BBM components which realize:

```
a = 3x + 2y (mod 2, mod p)
b = 2x + 3y (mod 2, mod p).
```

This can be implemented as:

1. q := XOR(x, y);
2. a delayless shift of one bit left;
3. a conditional r := q or r := XOR(q, p); then
4. a := XOR(r, x) and b := XOR(r, y).

This is 32 XOR gates per byte-wide mixing, with an assumed 3 level gate delay.

These byte-wide mixings are applied across the 64 bytes to be mixed, 32 pairs at a time, in FFT-like patterns. Note that each of the BBM operations is separate and independent hardware, functioning in parallel with other hardware. After 6 mixing sub-layers, every one of the 64 input bytes is equally represented in each of the 64 output bytes.

A 64-Byte-Block Mixing Cipher

Consider the chip construction of a 64-byte mixing cipher: we will have fencing, mixing, fencing, mixing, and fencing. With a 64-byte-wide block, we will have 64 byte-wide tables across the block, and three substitution layers, for a total of 192 byte-substitutions. This represents 48K (BYTES) or 384kb (bits) of RAM, which is about 11 percent of a current generation 4mb static RAM.

Note that each RAM generation is four times the size of the previous generation, and we can probably expect two such generations by year 2000. If so, we will see a 64mb static RAM chip, and the "aggressive" 384kb mixing design described here would cover only about 1/2 of 1 percent of such a chip.

There are two mixing layers in this mixing cipher, and in each we have 6 FFT-like sub-layers of 32 BBM's each. Thus, we need 32 * 6 * 2 = 384 BBM's for a total of about 12k gates. But note that the 384k RAM elements dominate the 12k gates by a factor of 32, so in this "back of an envelope" analysis we can ignore the gates with little error.

The 64 individual byte substitutions in each fencing layer each function in parallel, as do the 32 BBM's in each mixing sub-layer. In total, we have 15 separate layers of components which function in parallel. We can group and pipeline as many stages as we need (at almost no cost) to make our clock rate, which might be 200MHz.

This would allow us to insert a new 64-byte block on every clock. Thus, the projected ciphering rate is 64 bytes per clock, times 8 bits per byte, times 200,000,000 clocks per second, for a total of 102,400,000,000 bits per second. This is 102 * 10**9 b/sec; in the US this is called 102 *billion* bits per second (12.8 GB/sec).

Getting data on and off a chip is a well known issue now being addressed in other contexts. For example, nine DRAM vendors are establishing a standard for a next-generation DRAM architecture to keep up with continued massive improvements in CPU performance. Both the "SyncLinc" and Rambus approach are expected to deliver 800 MB/sec in 1997-98, and Rambus expects to deliver 3-4 GB/sec by 2001. Being beyond this is good: We would clearly like to have our ciphering structures remain the non-limiting element somewhat beyond this near date.

Mixing Construction Implications

This design sketches the simple construction and massive performance resulting from an aggressive realization. Here, a 384kb RAM area investment ciphers 64-byte-wide blocks at 12.8 GB/sec.

But a wide variety of less aggressive design strategies are obviously available as well. In particular, three separate layers of substitution could be reduced to one 128kb layer used 3 times, at some loss of strength. Similarly, the two mixings would be reduced to one 192-BBM mixing, used twice. The resulting 128kb cipher (about 3 percent of a 4mb static RAM chip) is one-third the size of the larger construction, but is more complex and ciphers at less than one-third the larger rate.

Continuing the theme of smaller designs, within a mixing, the 6 sublayers might be reduced to a single sublayer of just 32 BBM's, used 6 times. Ultimately, we could re-use a single 256-byte substitution across the block, and a single 32-gate BBM as well, although such a design seems pretty minimal.

Since a 64 byte block generally requires no chaining, it also supports independent block ciphering in multiple hardware. And this allows us to easily multiply the throughput of one large block device (in whatever realization we choose) by whatever amount of hardware we can afford. We might, for example, choose a low component count realization, and then replicate that to make our data rate.

In a very similar way, additional speed is available in even *wider* blocks, provided only that correspondingly more devices (and, to a lesser extent, more mixing sub-layers) are used. These devices do not have to switch faster to produce greater throughput. Thus, in a mixing construction, *wider* blocks can actually provide linearly *greater* throughput (with a modest increase in latency), as well as greater strength.

It is even possible to think about a low-end "smart card" mixing cipher realization.

Terry Ritter, his current address, and his top page.

Last updated: 1997-03-27

Measured Boolean Function Nonlinearity in Mixing Cipher Constructions

A *Ciphers By Ritter* Page

Terry Ritter

Nonlinearity is the number of bits which must change in the truth table of a Boolean function to reach the closest affine function. Although we cannot measure the nonlinearity of a cipher of practical size, we *can* measure modest substitution tables and small block constructions. Since a random substitution table is the ideal model of a block cipher, we can measure many random tables and develop an ideal nonlinearity distribution for a small block size. We can then measure small block constructions and compare distributions. Experimental results show that Mixing constructions can produce nonlinearity distributions which are surprisingly close to the ideal. These experiments tend to contradict the claim that block ciphers which use linear mixing are inherently weak.

From: ritter@io.com (Terry Ritter)
Newsgroups: sci.crypt
Subject: Measured Nonlinearity in Mixing Constructions (LONG!)
Date: Mon, 22 Dec 1997 08:18:14 GMT
Lines: 586
Message-ID: <349e21f4.2151248@news.io.com>

MEASURED NONLINEARITY IN MIXING CONSTRUCTIONS

Terry Ritter
Ritter Software Engineering
http://www.io.com/~ritter/

1997-12-21

Abstract

Nonlinearity is the number of bits which must change in the truth table of a Boolean function to reach the closest affine function. Although we cannot measure the nonlinearity of a cipher of practical size, we *can* measure modest substitution tables and small block constructions. Since a random substitution table is the ideal model of a block cipher, we can measure many random tables and develop an ideal nonlinearity distribution for a small block size. We can then measure small block constructions and compare distributions. Experimental results show that Mixing constructions can produce nonlinearity distributions which are surprisingly close to the ideal. These experiments tend to contradict the claim that block ciphers which use linear mixing are inherently weak.

Introduction

The ideal block cipher is a keyed simple substitution table of sufficient size. Unfortunately, with 128-bit blocks, there would be 2**128 entries in that table, which is completely out of the question. So the modern block cipher is a *construction* intended to *simulate* a keyed substitution of the desired size. At issue is the effectiveness of the construction technology. One way to investigate this is by using Boolean function theory, since a substitution table can be considered a set of independent Boolean functions, one for each output bit.

A Boolean function produces a single-bit result for each possible combination of values from perhaps many Boolean variables. The *nonlinearity* of a Boolean function is the Hamming distance to the closest affine function [e.g., PIE88, PIE89, PIE89B]. That is, nonlinearity is the number of bits which must change in the truth table of a Boolean function to reach the closest affine function. If "linearity" is considered a significant cryptographic weakness, nonlinearity is an explicit measure of the *lack* of that weakness. That is, nonlinearity measures one form of cipher "strength."

For cryptographic purposes, it is desired to take the nonlinearity of a substitution table to be the *minimum* of the nonlinearity values for each output bit in that table. Nonlinearity is measured by forming the (one-bit-wide) truth table for a particular output bit, then performing a Fast Walsh-Hadamard Transform (FWT) on that array. Each result value is essentially a correlation count to a particular affine function, and the minimum distance is found by scanning the transform results.

In measuring nonlinearity, it is generally necessary to record the function result for each possible combination of input variables. If we have an 8-bit function, we must record and then transform 256 elements, and if we have a 16-bit function, we must record and transform 64K elements. Measuring large functions rapidly becomes impossible. Although we cannot hope to measure the nonlinearity of a real 64-bit or 128-bit block cipher, we *can* measure nonlinearity in substitution tables and small block constructions. Of course, this is only reasonable for *scalable* constructions which can build both real ciphers and measurable tiny versions from the exact same plans.

Ideal Nonlinearity Distributions

By properly shuffling tables using a large-state cryptographic RNG (in particular:

http://www.io.com/~ritter/KEYSHUF.HTM

) we can sample a presumably uniform distribution of different table arrangements. We can measure the nonlinearity of each of these tables, and accumulate the results. For 4-bit tables, we get something like this:

```
  Nonlinearity Distribution in 4-Bit Tables
            Fig. 1
   0.6 |
   0.5 |     *  *
   0.4 |     *  *
   0.3 |     *  *
   0.2 |     *  *
   0.1 |     *  *
   0.0 |  *  *  *
  Prob +--+--+--+--
          0  2  4   Nonlinearity
```

First, note that nonlinearity always takes even values. Next, 4-bit invertible substitutions are surprisingly weak: Despite having 16 bits of Boolean function description, no 4-bit table was found to be more than 4 bits away from an affine function, and almost half were only 2 bits away. A few of these tables (probability 0.01) even contain a true affine function.

The 8-bit tables have the opportunity to be far more complex, since there are 256 bits in each 8-bit Boolean function:

```
  Nonlinearity Distribution in 8-Bit Tables
               Fig. 2
  0.35  |                 *
  0.3   |                 *
  0.25  |              *  *
  0.2   |              *  *  *
  0.15  |              *  *  *
  0.1   |           *  *  *  *
  0.05  |        *  *  *  *  *
  0.00  |     *  *  *  *  *  *  *
  Prob  +--+--+--+--+--+--+--+--+--
             92    96    100   104   Nonlinearity
```

Some of the literature asserts that *random* S-Boxes will have various good qualities provided the tables are "large enough" [AY82, GOR82, OC91, OC93, YOU95B]. The obvious question is: "How large is 'large enough'?" Experimental measurement of the nonlinearity of 1,000,000 random 8-bit tables shows exactly one table with a nonlinearity as low as 78; this is a 0.000001 probability (1 count in 1E6). So not only are random 8-bit tables with even a single affine coordinate (nonlinearity = 0) *unlikely*, they are *so* unlikely as to be almost impossible to find. (Indeed, [GOR82] gives the probability as 10**-72.) So accidental "linearity" in random 8-bit tables seems to be a non-issue.

Our block cipher construction problem essentially involves using small random tables (here 4-bit tables with 16 entries each and a mean nonlinearity of 3.0) to somehow emulate a table which is at least twice as wide (here an 8-bit table of 256 entries and a mean nonlinearity of 99.0). Stated in this way, a solution seems quite unlikely, for how can just 4, 6, or 8 tables of nonlinearity 3 ever produce a nonlinearity of 99? (We also examine using 5-bit tables with 32 entries and a mean nonlinearity of 7.8 emulating a 10-bit table with 1024 entries and a mean nonlinearity of 447.7, which seems similarly improbable.)

Mixing Ciphers

A mixing cipher (see, for example:

http://www.io.com/~ritter/CRYPHTML.HTM#MixTech

especially:

http://www.io.com/~ritter/MIXCORE.HTM

and:

http://www.io.com/~ritter/EXTRMSPD.HTM

) is distinguished by its use of high quality mixing, as opposed to repeated rounds of lesser mixing. Mixing ciphers generally use multiple substitution tables, each of which is shuffled for primary keying. The mixing itself is generally linear, but assures that each and every input affects each and every output in a balanced way. Presumably, nonlinearity measurements will have something to say about possible ill effects of this linear mixing.

With the particular mixing function which I call "Balanced Block Mixing," (see, for example:

http://www.io.com/~ritter/CRYPHTML.HTM#BBMTech

and especially:

http://www.io.com/~ritter/BBM.HTM

), a change in even one input is *guaranteed* to change every output, which is the basis for good overall diffusion and for reasoning about such diffusion. Mixing ciphers thus satisfy "the important cryptographic property of completeness" [HEY95].

This mixing is linear, and, by itself, has no strength at all. But the purpose of the mixing is to combine two input values into two output values, *each* of which depends upon *both* input values. The issue here is whether this sort of structure can produce higher nonlinearity values than the small substitution tables which are the only nonlinear components. Obviously, we would like to see the result approach the ideal nonlinearity distribution for the larger block.

Fig. 3 shows the basic mixing construction: The input block enters at the top, is split into two and each half substituted. Those results are mixed into two new block values, which are then substituted again. In this way the larger block is emulated by operations on half-width blocks. Each additional mixing means a new layer of mixing and a new layer of substitution.

```
     A Single Mixing    Fig. 3

         |     |
        SUB   SUB
          \   /
           MIX
          /   \
        SUB   SUB
         |     |
```

With a single mixing (and four random 4-bit tables) we find nonlinearity values only every 4 steps, instead of every 2:

```
  Nonlinearity Distribution with One Mixing Level
                    Fig. 4
  0.45  |                                 *
  0.4   |                                 *
  0.35  |                                 *
  0.3   |                                 *
  0.25  |                           *     *
  0.2   |                           *     *
  0.15  |                           *  *  *  *
  0.1   |                           *  *  *  *
  0.05  |                           *  *  *  *
  0.00  |  *           *  *  *      *  *  *  *
  Prob  +--+--+--+--+--+--+--+--+--+--+--+--+--+--
          56    64    72    80    88    96    100  Nonlinearity
```

This is a poor approximation to the ideal 8-bit distribution.

But with two mixings (and six random 4-bit tables) we can do much better (this represents one run of 10,000 tiny ciphers):

```
  Nonlinearity Distribution with Two Mixing Levels
               Fig. 5
  0.35  |                >*
  0.3   |                 *
  0.25  |             >*  *
  0.2   |              *  * >*
  0.15  |           *  *  *  *
  0.1   |          >*  *  *  *
  0.05  |       >*  *  *  *  *
  0.00  |  * >*  *  *  *  *  * >*
  Prob  +->+--+--+--+--+--+--+--+--
             92    96    100   104   Nonlinearity
```

Here the ">" symbols represent the ideal distribution.

The goal here seems to be not necessarily having the exact ideal distribution, but instead to gain confidence that the construction does produce reasonable *approximation* to the ideal. Indeed, one would expect that a random table must *inevitably* be qualitatively different from any possible construction which has a smaller total state. But Fig. 5 does seem to be a reasonable approximation, especially in view of Fig. 4.

Alas, chi-square comparisons are more exacting:

Chi-Square For Two 4-Bit Mixings   Fig. 6

NL	Expect	Found	Chi-Sq
106	2	1	}
104	249	242	} 0.255
102	2027	1847	16.984
100	3412	3256	7.132
98	2474	2520	0.855
96	1172	1306	15.321
94	450	530	14.222
92	150	182	6.827
90	46	65	7.848
88	13	29	} 60.500
86	4	11	}
84	1	7	}
82	0	3	}
80	0	1	}

This is obviously an unreasonable chi-square value. But if we add just one more mixing, for a total of 3 mixings and eight random 4-bit tables, we can do much better:

Chi-Square For Three 4-Bit Mixings   Fig. 7

NL	Expect	Found	Chi-Sq
106	2	2	}
104	249	256	} 0.195
102	2027	1985	0.870
100	3412	3386	0.198
98	2474	2518	0.855
96	1172	1179	0.042
94	450	446	0.036
92	150	175	4.167
90	46	35	2.630
88	13	17	} 0.000
86	4	1	}
84	1	0	}

			8.933

With 9 "degrees of freedom," this is a *reasonable* chi-square value, making the trial an arguable statistical match to the ideal distribution. So if an accurate distribution is in fact needed at this level, using the very weak 4-bit table components and only two tables across, three mixing levels would be preferred.

But this brings up two further questions: First, might two mixings be enough with 4-bit tables if we have 4 tables across? And, might two mixings be enough if we use 5-bit tables? We investigate the last question first, because it involves far less computation, and if the answer is positive, there is no practical need to investigate the first: Any serious cipher would use tables wider than 5-bits anyway.

First, we have the typical nonlinearity of 5-bit tables:

```
  Nonlinearity Distribution in 5-Bit Tables
            Fig. 8
   0.7 |              *
   0.6 |              *
   0.5 |              *
   0.4 |              *
   0.3 |              *
   0.2 |           *  *
   0.1 |           *  *  *
   0.0 |           *  *  *
  Prob +--+--+--+--+--+--+--
          0  2  4  6  8 10   Nonlinearity
```

Then the approximate ideal distribution for 10-bit tables:

```
  Nonlinearity Distribution in 10-Bit Tables
                      Fig. 9
  0.2   |
  0.175 |                    *  *
  0.15  |                    *  *  *
  0.125 |                 *  *  *  *
  0.1   |              *  *  *  *  *
  0.075 |              *  *  *  *  *  *
  0.05  |        *  *  *  *  *  *  *  *
  0.025 |     *  *  *  *  *  *  *  *  *  *
  0.00  |  *  *  *  *  *  *  *  *  *  *  *
  Prob  +--+--+--+--+--+--+--+--+--+--+--+--+--
          436   440   444   448   452   456   Nonlinearity
```

While this gives a general idea of the distribution, we now prefer to use the actual counts, scaled to the expectations of a smaller trial. The ideal model is thus constructed from three different 15-hour trials of 1,000,000 random 10-bit tables each, with the counts reduced by a factor of 100 and rounded to an integer.

The Ideal 10-Bit Nonlinearity Distribution   Fig. 10

NL	Trial 1	Trial 2	Trial 3	Ideal
462	0	1	0	0
460	55	75	55	1
458	2145	2269	2234	22
456	20537	20938	20882	208
454	76259	76344	75980	763
452	148847	148135	148510	1485
450	188329	187802	187585	1878
448	177775	178208	178616	1782
446	140283	140219	140407	1403
444	97415	97317	97465	974
442	61999	62282	62213	622
440	37829	37685	37382	376
438	21675	21789	21753	218
436	12417	12320	12408	124
434	6826	6818	6912	68
432	3740	3693	3584	37
430	1912	2050	1884	19
428	979	1043	1073	10
426	490	537	526	5
424	249	241	265	2
422	119	133	143	1
420	64	55	69	0
418	26	25	30	0
416	12	14	12	0
414	9	5	6	0
412	3	2	4	0
410	0	0	1	0

Next, we have nonlinearity measurements for 1,000 different 10-bit ciphers:

Chi-Square For Two 5-Bit Mixings   Fig. 11

NL	Expect	Found	Chi-Sq
458	2	3	}
456	21	13	} 2.333
454	76	74	0.053
452	149	139	0.667
450	188	202	1.043
448	178	198	2.247
446	140	125	1.607
444	97	95	0.041
442	62	62	0.000
440	38	36	0.105
438	22	31	3.682
436	12	10	0.333
434	7	6	} 0.286
432	4	4	}
430	2	0	}
428	1	2	}

			12.397

With 11 "degrees of freedom," this is a believable chi-square value. So we proceed with a 10,000-cipher trial:

Chi-Square For Two 5-Bit Mixings   Fig. 12

NL	Expect	Found	Chi-Sq
460	1	1	}
458	22	20	} 0.043
456	208	190	1.558
454	763	722	2.203
452	1485	1487	0.003
450	1878	1863	0.120
448	1782	1780	0.002
446	1403	1453	1.782
444	974	961	0.174
442	622	618	0.026
440	376	378	0.011
438	218	243	2.867
436	124	137	1.363
434	68	63	0.368
432	37	45	1.730
430	19	17	0.211
428	10	8	} 0.474
426	5	10	}
424	2	3	}
422	1	0	}
420	1	1	}

			12.935

With 15 "degrees of freedom," this is also a reasonable value. So 26 more 10,000-cipher trials were performed, with the following chi-square results:

Chi-Square Values for 10,000-Cipher Trials   Fig. 13

6.768	7.984	8.054	9.007	11.247	12.225
12.464	12.583	12.901	14.293	15.170	15.876
16.458	16.848	17.842	19.691	19.973	20.375
20.571	21.956	23.108	24.753	24.773	25.709
28.069	33.837				

The last value seems unreasonably high; either we got very unlucky, or the data are telling us that the match is not perfect. Perhaps we are seeing imperfections in the 5-bit tables the way we earlier saw problems in the 4-bit tables. But most of the values are very reasonable, and values like this do not repeatedly occur by chance alone.

Conclusions

These initial experiments support an assertion that a two-level Mixing cipher using tables of reasonable size can have a nonlinearity distribution unexpectedly similar to the ideal block cipher. The match seems sufficiently precise as to imply the existence of an underlying mathematical relationship which is currently unknown.

We conclude that Mixing constructions do in fact produce nonlinearity levels and distributions similar to those of an ideal cipher, despite using much smaller and much weaker components. These experiments thus support the Mixing approach to block cipher design, and do *not* support the claim that ciphers with linear mixing are inherently weak.

Note that the ability to perform these experiments is based on the "scalability" of the cipher design. This is an important advantage that very few designs offer: Scalability supports far deeper experimental analysis than is possible in a full-size cipher.

The approach to block cipher analysis taken in this article seems to have not been reported previously in the open literature.

References and Bibliography

[AY82] Ayoub, F. 1982. Probabilistic completeness of substitution-permutation encryption networks. IEE Proceedings, Part E. 129(5): 195-199.

[DAE94] Daemen, J., R. Govaerts and J. Vandewalle. 1994. Correlation Matrices. Fast Software Encryption. 275-285.

[FOR88] Forre, R. 1988. The Strict Avalanche Criterion: Spectral Properties of Boolean Functions and an Extended Definition. Advances in Cryptology -- CRYPTO '88. 450-468.

[GOR82] Gordon, J. and H. Retkin. 1982. Are Big S-Boxes Best? Cryptography. Proceedings of the Workshop on Cryptography, Burg Feuerstein, Germany, March 29-April 2, 1982. 257-262.

[HEY94] Heys, H. and S. Tavares. 1994. On the security of the CAST encryption algorithm. Canadian Conference on Electrical and Computer Engineering. Halifax, Nova Scotia, Canada, Sept. 1994. 332-335.

[HEY95] Heys, H. and S. Tavares. 1995. Known plaintext cryptanalysis of tree-structured block ciphers. Electronics Letters. 31(10): 784-785.

[MEI89] Meier, W. and O. Staffelbach. 1989. Nonlinearity Criteria for Cryptographic Functions. Advances in Cryptology -- Eurocrypt '89. 549-562.

[MIR97] Mirza, F. 1997. Linear and S-Box Pairs Cryptanalysis of the Data Encryption Standard.

[OC91] O'Connor, L. 1991. Enumerating nondegenerate permutations. Advances in Cryptology -- Eurocrypt '91. 368-377.

[OC93] O'Connor, L. 1993. On the Distribution Characteristics in Bijective Mappings. Advances in Cryptology -- EUROCRYPT '93. 360-370.

[PIE88] Pieprzyk, J. and G. Finkelstein. 1988. Towards effective nonlinear cryptosystem design. IEE Proceedings, Part E. 135(6): 325-335.

[PIE89] Pieprzyk, J. and G. Finkelstein. 1989. Permutations that Maximize Non-Linearity and Their Cryptographic Significance. Computer Security in the Age of Information. 63-74.

[PIE89B] Pieprzyk, J. 1989. Non-linearity of Exponent Permutations. Advances in Cryptology -- EUROCRYPT '89. 80-92.

[PIE93] Pieprzyk, J., C. Charnes and J. Seberry. 1993. Linear Approximation Versus Nonlinearity. Proceedings of the Workshop on Selected Areas in Cryptography (SAC '94). 82-89.

[PRE90] Preneel, B., W. Van Leekwijck, L. Van Linden, R. Govaerts and J. Vandewalle. 1990. Propagation Characteristics of Boolean Functions. Advances in Cryptology -- Eurocrypt '90. 161-173.

[RUE86] Rueppel, R. 1986. Analysis and Design of Stream Ciphers. Springer-Verlag.

[XIO88] Xiao, G-Z. and J. Massey. 1988. A Spectral Characterization of Correlation-Immune Combining Functions. IEEE Transactions on Information Theory. 34(3): 569-571.

[YOU95] Youssef, A. and S. Tavares. 1995. Resistance of Balanced S-boxes to Linear and Differential Cryptanalysis. Information Processing Letters. 56: 249-252.

[YOU95B] Youssef, A. and S. Tavares. 1995. Number of Nonlinear Regular S-boxes. Electronics Letters. 31(19): 1643-1644.

[ZHA95] Zhang, X. and Y. Zheng. 1995. GAC -- the Criterion for Global Avalanche Characteristics of Cryptographic Functions. Journal for Universal Computer Science. 1(5): 316-333.

Terry Ritter   ritter@io.com   http://www.io.com/~ritter/
The Crypto Glossary: http://www.io.com/~ritter/GLOSSARY.HTM

Also see: **Measuring Nonlinearity by Walsh Transform (1998)** (20K)

Terry Ritter, his current address, and his top page.

Last updated: 1997-12-28

Measured Boolean Function Nonlinearity in Variable Size Block Ciphers

A *Ciphers By Ritter* Page

Terry Ritter

Nonlinearity is the number of bits which must change in the truth table of a Boolean function to reach the closest affine function, and thus is a measure of one kind of cipher strength. Although we cannot measure the nonlinearity of a cipher of practical size, we *can* measure modest substitution tables and small block constructions. Since a random substitution table is the ideal model of a block cipher, we can measure many random tables and develop an ideal nonlinearity distribution. We can then measure block constructions of the same size and compare distributions. Experimental results indicate that VSBC constructions can produce nonlinearity distributions which are surprisingly close to the ideal. These experiments tend to contradict the claim that block ciphers which use VSBC constructions are inherently weak.

From: ritter@io.com (Terry Ritter)
Newsgroups: sci.crypt
Subject: Measured Nonlinearity in Variable Size Block Ciphers (LONG!)
Date: Sun, 18 Jan 1998 06:14:36 GMT
Lines: 690
Message-ID: <34c19dc1.27169161@news.io.com>

MEASURED NONLINEARITY IN VARIABLE SIZE BLOCK CIPHERS

Terry Ritter
Ritter Software Engineering
http://www.io.com/~ritter/

1998-01-17

Abstract

Nonlinearity is the number of bits which must change in the truth table of a Boolean function to reach the closest affine function, and thus is a measure of one kind of cipher strength. Although we cannot measure the nonlinearity of a cipher of practical size, we *can* measure modest substitution tables and small block constructions. Since a random substitution table is the ideal model of a block cipher, we can measure many random tables and develop an ideal nonlinearity distribution. We can then measure block constructions of the same size and compare distributions. Experimental results indicate that VSBC constructions can produce nonlinearity distributions which are surprisingly close to the ideal. These experiments tend to contradict the claim that block ciphers which use VSBC constructions are inherently weak.

Introduction

The ideal block cipher is a keyed simple substitution table of sufficient size. Unfortunately, with 128-bit blocks, there would be 2**128 entries in that table, which is completely out of the question. So the modern block cipher is a *construction* intended to *simulate* a keyed substitution of the desired size. At issue is the effectiveness of the construction technology. One way to investigate this is by using Boolean function theory, since a substitution table -- or cipher -- can be considered a set of independent Boolean functions, one for each output bit.

A Boolean function produces a single-bit result for each possible combination of values from perhaps many Boolean variables. The *nonlinearity* of a Boolean function is the Hamming distance to the closest affine function [e.g., PIE88, PIE89, PIE89B]. (Also see "Measuring Nonlinearity by Walsh Transform" by this author:

http://www.io.com/~ritter/ARTS/MEASNONL.HTM

.) That is, nonlinearity is the number of bits which must change in the truth table of a Boolean function to reach the closest affine function. If "linearity" is considered a significant cryptographic weakness, nonlinearity is an explicit measure of the *lack* of that weakness. So nonlinearity measures one form of cipher "strength."

For cryptographic purposes, it is desired to take the nonlinearity of a substitution table to be the *minimum* of the nonlinearity values for each output bit in that table. Nonlinearity is measured by forming the (one-bit-wide) truth table for a particular output bit, then performing a Fast Walsh-Hadamard Transform (FWT) on that array. Each result value is essentially a correlation count to a particular affine function, and the minimum distance is found by scanning the transform results.

In measuring nonlinearity, it is generally necessary to record the function result for each possible combination of input variables. If we have an "8-bit" table, we must record and then transform 256 elements, and if we have a "16-bit" table, we must record and transform 64K elements. Thus, measuring large functions rapidly becomes impossible. So, although we cannot hope to measure the nonlinearity of a real 64-bit or 128-bit block cipher, we *can* measure nonlinearity in substitution tables and small block constructions.

Ideal Nonlinearity Distributions

By properly shuffling tables using a large-state cryptographic RNG (in particular:

http://www.io.com/~ritter/KEYSHUF.HTM

) we can sample a presumably uniform distribution of different table arrangements. We can measure the nonlinearity of each of these tables, and accumulate the results. For 4-bit tables, we get something like that shown in Figure 1.

```
  Nonlinearity Distribution in 4-Bit Tables    Fig. 1

   0.6 |
   0.5 |     *  *
   0.4 |     *  *
   0.3 |     *  *
   0.2 |     *  *
   0.1 |     *  *
   0.0 |  *  *  *
  Prob +--+--+--+--
          0  2  4   Nonlinearity
```

In balanced Boolean functions, nonlinearity always takes even values: If we have two balanced functions at some distance from each other and change one bit in the second function, the distance between them will change by one bit, and the second function will also no longer be balanced. To balance the second function, we have to change *another* bit, which either adds to the original change, or cancels it out. In the end, the distance between the functions has changed by +2, -2, or 0 bits.

The 4-bit invertible substitutions are surprisingly weak: Despite having 16 bits of Boolean function description, no 4-bit table was found to be more than 4 bits away from an affine function, and almost half were only 2 bits away. A few of these tables (probability 0.01) even contained a true affine function.

In contrast, the 8-bit tables have the opportunity to be far more complex, since there are 256 bits in each 8-bit Boolean function. This is shown in Figure 2.

```
  Nonlinearity Distribution in 8-Bit Tables    Fig. 2

  0.35  |                 *
  0.3   |                 *
  0.25  |              *  *
  0.2   |              *  *  *
  0.15  |              *  *  *
  0.1   |           *  *  *  *
  0.05  |        *  *  *  *  *
  0.00  |     *  *  *  *  *  *  *
  Prob  +--+--+--+--+--+--+--+--+--
             92    96    100   104   Nonlinearity
```

Some of the literature asserts that *random* S-Boxes will have various good qualities provided the tables are "large enough" [AY82, GOR82, OC91, OC93, YOU95B]. This leads to some anxiety over the question of "How large is 'large enough'?" In our experiments, measurement of the nonlinearity of 1,000,000 random 8-bit tables found exactly one table with a nonlinearity as low as 78; this is a 0.000001 probability (1 count in 1E6). So not only are random 8-bit tables with even a single affine coordinate (nonlinearity = 0) *unlikely*, they are *so* unlikely as to be almost impossible to find. (Indeed, [GOR82] gives the probability as 10**-72.) So finding accidental "linearity" in random 8-bit tables seems to be an issue only in theory: this is a non-issue in practice.

Our block cipher construction problem essentially involves using small random tables (here 4-bit tables with 16 entries each and a mean nonlinearity of 3.0) to somehow emulate a table which is three times as wide (here a 12-bit table of 4096 entries and a mean nonlinearity of 1907.9). Stated in this way, a solution seems quite unlikely, for how can even 15 tables of nonlinearity 3 ever produce a nonlinearity of 1908?

Variable Size Block Ciphers

A Variable Size Block Cipher (VSBC) (see, for example:

http://www.io.com/~ritter/CRYPHTML.HTM#VSBCTech

and especially:

http://www.io.com/~ritter/VSBCCORE.HTM

) differs from most other ciphers in its ability to handle blocks of dynamically arbitrary size to the byte. This is accomplished with one-way mixing layers (see Figure 3) which guarantee that any change to any input is reflected in all subsequent mixed bytes. By using multiple such layers we can guarantee that any single-bit change in the data will affect all columns of the ciphertext result.

```
  A One-Way Variable-Size Mixing Layer    Fig. 3

        |     |     |             |
        |     |     |             |
        +--->MIX-->MIX--> ... -->MIX----+
              |     |             |     |
              v     v             v     v
```

Figure 3 shows a single one-way mixing layer: Each element value (in practice, each element is typically a byte) enters at the top, and the mixed results flow out the bottom. The typical Variable Size Block Cipher uses multiple layers of mixing, usually in opposite directions, to produce high quality overall mixing in a fixed-depth structure.

VSBC's generally use multiple substitution tables, each of which is shuffled for primary keying. (See again, for example:

http://www.io.com/~ritter/KEYSHUF.HTM

.) The issue here is whether this sort of structure can produce higher nonlinearity values than the small substitution tables which are the only nonlinear components. Obviously, we would like to see the result approach the ideal nonlinearity distribution for the larger block.

Figure 4 shows a two-element VSBC construction: The input block enters at the top, is split into two and each half substituted. Those results are mixed into two new block values; this is an unusual invertible mixing which we call Balanced Block Mixing (see, for example:

http://www.io.com/~ritter/CRYPHTML.HTM#BBMTech

and especially:

http://www.io.com/~ritter/BBM.HTM

). After mixing, the results are substituted again. There is another mixing, another substitution and the two-element block flows out the bottom.

```
    A Two-Column Toy VSBC    Fig. 4

        |     |
       SUB   SUB
        |     |
        +--->MIX----+
              |     |
             SUB   SUB
              |     |
        +----MIX<---+
        |     |
       SUB   SUB
        |     |
        v     v
```

We can compare the model of Figure 4 to the Mixing structure investigated previously (see

http://www.io.com/~ritter/ARTS/MIXNONLI.HTM

), as shown in Figure 5, and see that they turn out to be the same, when reduced to this minimal structure.

```
   A Two-Element Mixing Cipher   Fig. 5

         |     |
        SUB   SUB
          \   /
           MIX
          /   \
        SUB   SUB
          \   /
           MIX
          /   \
        SUB   SUB
         |     |
```

The identity between the two-element-width Mixing cipher and the toy VSBC of similar width means that the results from the earlier investigation apply here:

1. A 2-element-width construction with 3 mixing levels, using 8 random 4-bit substitution tables, does appear to produce a nonlinearity distribution consistent with the ideal for an 8-bit table.

2. A 2-element-width construction with 2 mixing levels, using 6 random 5-bit substitution tables, does appear to produce a nonlinearity distribution consistent with the ideal for a 10-bit table.

On the other hand, the earlier results also make clear the inherent weakness in 4-bit tables: A 2-element-width construction with just 2 mixing levels, using 6 random 4-bit tables (the structure shown in Figure 5) was clearly *not* ideal. And in the current investigation we *must* use 4-bit tables if we are to have a reasonable amount of desktop computation.

Here we investigate whether a 12-bit VSBC will produce something like the nonlinearity distribution of the ideal 12-bit cipher. So first we need to identify the ideal 12-bit nonlinearity distribution.

The 12-Bit Ideal

The ideal 12-bit nonlinearity distribution was constructed from three different 12-hour trials of 100,000 random 12-bit tables each. (We really would like to have trials at least 10 times this size, but that would mean three 5-day runs for me, so the current result should be thought of as a decent guess.) Each ideal value is near the average of the three trials, weighted toward any two similar values, with the counts reduced by a factor of 10 and rounded to an integer, as shown in Figure 6.

The Ideal 12-Bit Nonlinearity Distribution   Fig. 6

NL	Trial 1	Trial 2	Trial 3	Ideal
1838	1	0	0	0
1840	0	0	0	0
1842	0	0	0	0
1844	0	0	2	0
1846	3	0	0	0
1848	0	0	2	0
1850	3	0	0	0
1852	2	2	1	0
1854	2	2	2	0
1856	2	4	2	0
1858	2	0	4	1
1860	10	14	12	0
1862	14	10	6	1
1864	16	11	14	2
1866	32	28	17	3
1868	29	28	32	3
1870	27	52	58	5
1872	68	64	66	7
1874	92	84	98	9
1876	119	134	127	12
1878	185	162	197	19
1880	253	247	249	25
1882	323	353	312	32
1884	488	467	507	48
1886	652	643	674	65
1888	926	885	915	91
1890	1152	1185	1207	118
1892	1506	1566	1660	157
1894	2094	2132	2108	211
1896	2762	2772	2751	276
1898	3711	3643	3552	364
1900	4596	4572	4665	460
1902	5614	5753	5624	563
1904	6974	7050	7022	702
1906	8158	8210	8136	816
1908	9395	9489	9313	939
1910	10155	10101	10132	1013
1912	10414	10276	10306	1032
1914	9584	9482	9693	958
1916	8092	8062	8105	809
1918	6020	5915	5866	593
1920	3701	3701	3715	370
1922	1855	1903	1839	186
1924	697	753	766	74
1926	214	201	201	20
1928	48	39	40	4
1930	8	5	3	1
1932	1	0	0	0

The Toy Model

A reasonable first step to investigating VSBC nonlinearity is to extend the 2-element structure of Figure 4 into the 3-element structure shown in Figure 7.

```
   A Three-Column Toy Variable-Size Block Cipher    Fig. 7

        |     |     |
       SUB   SUB   SUB
        |     |     |
        +--->MIX-->MIX----+
              |     |     |
             SUB   SUB   SUB
              |     |     |
        +----MIX<--MIX<---+
        |     |     |
       SUB   SUB   SUB
        |     |     |
        v     v     v
```

Here, for the first time, the one-way characteristic of the VSBC mixing layers becomes apparent. In the figure, three 4-bit elements enter the top, are substituted and mixed, and the resulting ciphertext flows out the bottom. The overall result is a keyed permutation; a simulated large simple substitution table.

Toy Model Test

In each trial, we first initialize all the substitutions from a random key. This produces a simulated 12-bit simple substitution which we then measure for nonlinearity. At first we will perform just a small trial of 1,000 different 12-bit VSBC initializations, and this result is shown in Figure 8.

Chi-Square For 12-Bit VSBC   Fig. 8

NL	Expect	Found
1408	0	1
1472	0	2
1536	0	10
1568	0	32
1600	0	52
1632	0	90
1664	0	164
1696	0	216
1712	0	1
1728	0	215
1736	0	1
1756	0	1
1760	0	150
1788	0	1
1792	0	56
1824	0	8

The figure shows the results of measuring 1,000 randomly-keyed constructions, and *not* *one* of these values is as high as the *minimum* value (typically 1850) we might expect from the ideal 12-bit distribution. These results are so far from the expectation as to scarcely require computation: a computed chi-square total would be something like 100,000. Thus, the toy VSBC model of Figure 7 does *not* even *begin* to approach the distribution we want from an ideal cipher.

The Full Model

It is known from previous work that a two-level VSBC mixing is weak, so the next step is to extend the structure of Figure 7 to have 2 one-way mixing levels in each direction, as shown in Figure 9. Again, we must use the weak 4-bit tables to make the computation reasonable.

```
  A Three-Column Variable-Size Block Cipher    Fig. 9

        |     |     |
       SUB   SUB   SUB
        |     |     |
        +--->MIX-->MIX----+
              |     |     |
             SUB   SUB   SUB
              |     |     |
              +--->MIX-->MIX----+
                    |     |     |
                   SUB   SUB   SUB
                    |     |     |
              +----MIX<--MIX<---+
              |     |     |
             SUB   SUB   SUB
              |     |     |
        +----MIX<--MIX<---+
        |     |     |
       SUB   SUB   SUB
        |     |     |
        v     v     v
```

As before, three 4-bit elements enter the top, are substituted and mixed multiple times, and the resulting ciphertext flows out the bottom.

(In practice, a working design would use the vastly stronger 8-bit tables, and would also use dynamic table selection, so that the same tables would only rarely occur in the same ciphering position. But neither of those improvements are used in these tests.)

Full Model Tests

In each trial, we first initialize all the substitutions from a random key. This produces a particular 12-bit cipher -- a simulated 12-bit simple substitution. We then traverse the complete transformation -- all 4096 possible data values -- and collect all 4096 12-bit ciphertext results. We then traverse all 12 ciphertext bit-columns, one-by-one, and perform a nonlinearity computation. The minimum nonlinearity value found across all columns is then accumulated into the growing distribution. Here we will perform full trials of 10,000 different 12-bit VSBC initializations each, with one such trial shown in Figure 10.

Chi-Square For 12-Bit VSBC   Fig. 10

NL	Expect	Found	Chi-Sq	DF
1850	0	0	}	
1852	0	2	}	
1854	0	0	}	
1856	0	0	}	
1858	1	2	}	
1860	0	0	}	
1862	1	0	}	
1864	2	1	}	
1866	3	4	}	
1868	3	2	} 0.100	0
1870	5	7	}	
1872	7	5	} 0.100	1
1874	9	5	}	
1876	12	16	} 0.100	2
1878	19	28	4.363	3
1880	25	25	4.363	4
1882	32	46	10.488	5
1884	48	42	11.238	6
1886	65	85	17.392	7
1888	91	77	19.546	8
1890	118	126	20.088	9
1892	157	153	20.190	10
1894	211	214	20.233	11
1896	276	263	20.845	12
1898	364	346	21.735	13
1900	460	453	21.842	14
1902	563	576	22.142	15
1904	702	719	22.554	16
1906	816	813	22.565	17
1908	939	947	22.633	18
1910	1013	1030	22.918	19
1912	1032	973	26.291	20
1914	958	967	26.376	21
1916	809	816	26.436	22
1918	593	588	26.478	23
1920	370	362	26.651	24
1922	186	199	27.560	25
1924	74	81	28.222	26
1926	20	24	} 28.382	27
1928	4	3	}	
1930	1	0	}	
1932	0	0	}	
1934	0	0	}	
1936	0	0	}	
1938	0	0	}	

With 27 "degrees of freedom," the chi-square value of 28.382 found in Figure 8 is very believable. The critical chi-square values for DF = 27 are shown in Figure 11.

Chi-Square Critical Values for DegFree = 27   Fig. 11

1%	5%	25%	50%	75%	95%	99%
12.878	16.151	21.749	26.336	31.528	40.113	46.962

Additional trials were performed (at 1.4 hours each), producing the overall chi-square results of Figure 12.

Chi-Square Values for 10,000-Cipher Trials   Fig. 12

11.362	13.933	17.418	17.822	19.520
20.052	20.954	20.992	21.024	21.084
21.415	21.936	22.629	22.913	24.171
24.224	24.291	25.306	26.263	26.781
27.754	28.382	28.619	29.252	30.271
31.369	31.388	32.555	33.015	33.367
33.990	34.883	35.674	36.171	37.987
39.853	39.980	43.197	45.514	

Most of the collected values seem reasonable.

One value *is* unexpectedly low, meaning that the sample distribution was unexpectedly *close* to the ideal. The actual probability of finding a chi-square value of 11.367 (by random sampling the ideal distribution) with 27 degrees of freedom is 0.0036. This is about 4 in 1000, so we would *expect* to get something like this if we would just conduct 278 trials. But random trials do not know how many trials have passed before, and the fact is that the value occurred early. There is no preponderance of such occurrences.

Also, in this data set, the tails seem to be represented more frequently than the center.

Despite these variations, all but one of the values seem at least reasonable, and -- as we saw previously -- chi-square values in this particular range do not repeatedly occur by chance alone. We are thus forced to conclude that the sampled distribution is reasonably close to the constructed ideal distribution for random 12-bit substitutions.

Conclusions

From these results we conclude that the 3-element VSBC construction using 15 random 4-bit tables does in fact produce nonlinearity levels and distributions *remarkably* *similar* to those of an ideal 12-bit cipher. These experiments thus support the VSBC approach to block cipher design, and do *not* support the claim that such ciphers are inherently weak.

Note that the ability to perform these experiments is based on the "scalability" of the cipher design. This is an important advantage that very few designs offer: Scalability supports far deeper experimental analysis than is possible in a full-size cipher.

I now believe that the inability of a cipher to support the deep analysis available to a scaled design is nothing less than a flaw in the cipher design itself. There is no theory of cipher strength such that, if we only follow those rules, we will build a "strong" cipher. Nor can we hope to thoroughly review any large cipher experimentally. It would seem that the only way we can hope to find completely unexpected strength problems is to *scale* the cipher to a size which we *can* address experimentally. Since large ciphers cannot be thoroughly reviewed, they also cannot be fully trusted.

References and Bibliography

[AY82] Ayoub, F. 1982. Probabilistic completeness of substitution-permutation encryption networks. IEE Proceedings, Part E. 129(5): 195-199.

[DAE94] Daemen, J., R. Govaerts and J. Vandewalle. 1994. Correlation Matrices. Fast Software Encryption. 275-285.

[FOR88] Forre, R. 1988. The Strict Avalanche Criterion: Spectral Properties of Boolean Functions and an Extended Definition. Advances in Cryptology -- CRYPTO '88. 450-468.

[GOR82] Gordon, J. and H. Retkin. 1982. Are Big S-Boxes Best? Cryptography. Proceedings of the Workshop on Cryptography, Burg Feuerstein, Germany, March 29-April 2, 1982. 257-262.

[HEY94] Heys, H. and S. Tavares. 1994. On the security of the CAST encryption algorithm. Canadian Conference on Electrical and Computer Engineering. Halifax, Nova Scotia, Canada, Sept. 1994. 332-335.

[HEY95] Heys, H. and S. Tavares. 1995. Known plaintext cryptanalysis of tree-structured block ciphers. Electronics Letters. 31(10): 784-785.

[MEI89] Meier, W. and O. Staffelbach. 1989. Nonlinearity Criteria for Cryptographic Functions. Advances in Cryptology -- Eurocrypt '89. 549-562.

[MIR97] Mirza, F. 1997. Linear and S-Box Pairs Cryptanalysis of the Data Encryption Standard.

[OC91] O'Connor, L. 1991. Enumerating nondegenerate permutations. Advances in Cryptology -- Eurocrypt '91. 368-377.

[OC93] O'Connor, L. 1993. On the Distribution Characteristics in Bijective Mappings. Advances in Cryptology -- EUROCRYPT '93. 360-370.

[PIE88] Pieprzyk, J. and G. Finkelstein. 1988. Towards effective nonlinear cryptosystem design. IEE Proceedings, Part E. 135(6): 325-335.

[PIE89] Pieprzyk, J. and G. Finkelstein. 1989. Permutations that Maximize Non-Linearity and Their Cryptographic Significance. Computer Security in the Age of Information. 63-74.

[PIE89B] Pieprzyk, J. 1989. Non-linearity of Exponent Permutations. Advances in Cryptology -- EUROCRYPT '89. 80-92.

[PIE93] Pieprzyk, J., C. Charnes and J. Seberry. 1993. Linear Approximation Versus Nonlinearity. Proceedings of the Workshop on Selected Areas in Cryptography (SAC '94). 82-89.

[PRE90] Preneel, B., W. Van Leekwijck, L. Van Linden, R. Govaerts and J. Vandewalle. 1990. Propagation Characteristics of Boolean Functions. Advances in Cryptology -- Eurocrypt '90. 161-173.

[RUE86] Rueppel, R. 1986. Analysis and Design of Stream Ciphers. Springer-Verlag.

[XIO88] Xiao, G-Z. and J. Massey. 1988. A Spectral Characterization of Correlation-Immune Combining Functions. IEEE Transactions on Information Theory. 34(3): 569-571.

[YOU95] Youssef, A. and S. Tavares. 1995. Resistance of Balanced S-boxes to Linear and Differential Cryptanalysis. Information Processing Letters. 56: 249-252.

[YOU95B] Youssef, A. and S. Tavares. 1995. Number of Nonlinear Regular S-boxes. Electronics Letters. 31(19): 1643-1644.

[ZHA95] Zhang, X. and Y. Zheng. 1995. GAC -- the Criterion for Global Avalanche Characteristics of Cryptographic Functions. Journal for Universal Computer Science. 1(5): 316-333.

Terry Ritter   ritter@io.com   http://www.io.com/~ritter/
The Crypto Glossary: http://www.io.com/~ritter/GLOSSARY.HTM

Also see: **Measuring Nonlinearity by Walsh Transform (1998)** (20K)
and: **Measured Nonlinearity in Mixing Constructions (1997)** (25K)

Terry Ritter, his current address, and his top page.

Last updated: 1998-02-26

Ritter's Cipher Boutique

Original architectures for custom ciphers.

Here you can select a base cipher and features for a custom cipher. Our existing cipher products can be modified for your needs, and our cipher prototypes can be expanded into full-blown custom ciphers in many different ways.

Contents

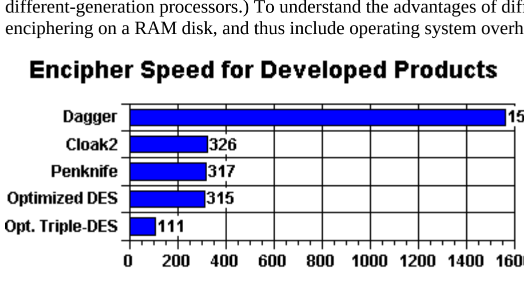
- Cipher Speed Discussion
- Cipher Strength Discussion
- Stream Ciphers
  - Dynamic Substitution
    - Penknife: originally an e-mail cipher with strong key-management facilities; for DOS or Microsoft Windows.
    - Cloak2: originally a file cipher with strong key-management facilities; for DOS or Microsoft Windows.
    - Dagger: originally a fast cipher engine for use by software developers.
- Block Ciphers
  - Fencing and Mixing Block Ciphers
    - Fenced DES: A 256-bit block cipher which uses DES as the main internal component.
    - Fenced Tree: A 256-bit block cipher which uses layers of increasingly fine mixing and three layers of keyed substitutions.
    - Fenced Quad: A 256-bit block cipher based on "FFT"-style mixing with the balanced block mixing component in two sizes.
    - Fenced FFT: A 256-bit block cipher based on increasingly fine "FFT"-style mixing with the balanced block mixing component.
    - Fenced OLS: A 256-bit block cipher based on "FFT"-style mixing using a orthogonal pair of latin squares.
  - Variable Size Block Ciphers (VSBC)
    - VSBC SubX: A Variable Size Block Cipher using substitution and exclusive-OR.
    - VSBC Ls4: A Variable Size Block Cipher based on Latin square combining.
    - VSBC LsX: A Variable Size Block Cipher using a Latin square and exclusive-OR.
    - VSBC SBBM: A Variable Size Block Cipher using substitution and balanced block mixing.
    - VSBC OLS: A Variable Size Block Cipher using an orthogonal pair of Latin square mixers.
- Summary

CIPHER SPEED

Within a given technology, a software cipher must to some extent trade off between speed and strength. But improved technology can provide better tradeoffs.

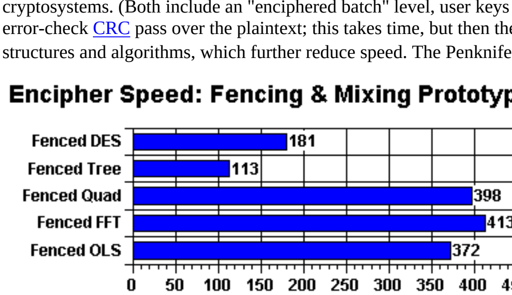
Ciphering speed is *very* important for any sort of centralized resource like LAN or network servers. Naturally, users who communicate with a LAN must use the cipher the LAN uses, even if cipher speed is not important to those users.

Speed is relatively easy to measure, but speed values vary with processor, clock speed, cache, operating system, code optimization, in-memory *vs.* disk ciphering, etc. (Even cycle-counts will differ on different-generation processors.) To understand the advantages of different techniques, we need to compare "apples with apples" in groups of similar "apples." These measurements are for file enciphering on a RAM disk, and thus include operating system overhead for file open, close, and data read/write.



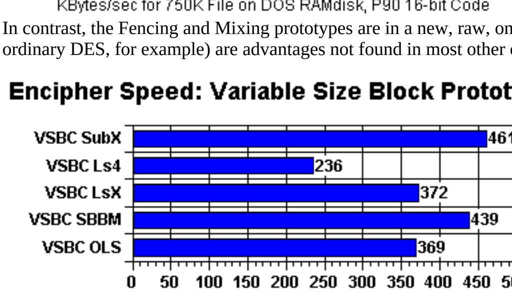
KBytes/sec for 750K File on DOS RAMdisk, P90 16-bit Code

Among our "more developed" implementations, DES, Triple-DES and Dagger are probably the most highly optimized. Penknife and Cloak2 are less optimal, partly because they are complex, full cryptosystems. (Both include an "enciphered batch" level, user keys in an alias file which initialize the data cipher, wildcard ciphering, etc.) For example, both Penknife and Cloak2 perform a separate error-check CRC pass over the plaintext; this takes time, but then the user knows when the file has been deciphered properly. Penknife also saves ciphertext as ASCII "lines," and has error-resilient data structures and algorithms, which further reduce speed. The Penknife and Cloak2 ciphers measure comparable to DES because real cryptosystems do more than just cipher data.



KBytes/sec for 750K File on DOS RAMdisk, P90 16-bit Code

In contrast, the Fencing and Mixing prototypes are in a new, raw, only partially optimized state. The ability to support large blocks, large keys, and ciphering based on existing ciphers (Fenced DES uses ordinary DES, for example) are advantages not found in most other ciphers. Again, this is actual file-to-file ciphering under DOS 6.21.



KBytes/sec for 750K File on DOS RAMdisk, P90 16-bit Code

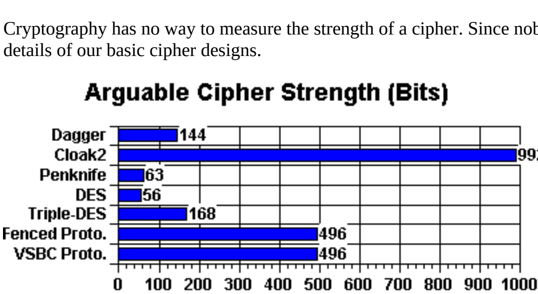
The Variable Size Block Ciphers are an even more unique category. Not only are these implementations new and raw, but the designs include features which are going to be used in a real system anyway. As measured, these ciphers include:

- actual file-to-file ciphering under DOS 6.21,
- a huge average 2K block size,
- a unique pseudo-random block size,
- extra "pre" and "post" padding blocks which hold no real data,
- an error-check field on every block,
- a separate validation value for every block, and
- some designs chain initialization values (IV) from block to block.

The VSBC designs do more, and would be better compared to packages with similar features, but some of these features simply will not be present in *any* other implementation.

CIPHER STRENGTH

Cryptography has no way to measure the strength of a cipher. Since nobody can *measure* cipher strength, we are left to *argue* strength. To provide an honest basis for discussion, we disclose the full details of our basic cipher designs.



Cipher "strength" would seem to be the entire reason for using a cipher, but here *more* is not necessarily *better*. Once beyond the effort which an Opponent can possibly apply, the strength is probably "good enough." The problem is that we can never really know how easy it will be to break a cipher, given advanced cryptoanalytic knowledge which will always be secret. A real cryptanalyst is not going to announce when a cipher is broken, for it is only then that the investment in the work starts paying off.

We can make the cryptography war more difficult for the cryptanalyst, by giving our organizations or groups their own custom cipher. Far more than simply adding a few bits to the keyspace, each custom cipher must be attacked separately. Since much less information will be protected by a custom cipher than, say, DES, there will be much less motive for such an attack, and fewer resources available for it. Custom ciphers give the cryptographer a way to fight against the advantage the cryptanalyst normally holds.

Right now, 56 bits is too weak for new designs, and at least 90 bits has recently been recommended. However, that group apparently did not discuss the effect that "molecular computation" could have in as little as five years. Accordingly, we would now like to see 112 or 120 bits as a reasonable *minimum* size key. (These are secret key sizes: A public key must be something like 10 times as large as a secret key for similar strength.) We tend to use much larger keys in our designs, because storing large keys is a minimal cost, and because large keys can make for straightforward and unambiguous designs.

Of course, no cipher can hide something which is not otherwise secret. The simple use of a cipher program will not, by itself, solve all data security problems.

STREAM CIPHERS

Stream ciphers deal with bits or bytes of the message at a time. In the Classical stream cipher, each byte is ciphered independent of any other byte. Modern forms cause each byte in the message to "affect" subsequent data bytes, but no true stream cipher can affect *earlier* bytes.

Basically, a stream cipher has two components: a *random number generator* (RNG), and a *combiner.* The RNG produces a *confusion sequence* which the combiner mixes with the *plaintext* message to produce *ciphertext.* The original plaintext is recovered by mixing the exact same confusion sequence with the ciphertext in an inverse combiner or *extractor.*

In our own designs we typically use an Additive RNG (see the 1991 RNG survey article (168K), especially Section 4.10) because we can make these very large and very fast. (We have also used conventional congruential and shift-register generators -- as in Penknife -- and tables -- as in Dagger.) We can vary the running size, the particular primitive polynomial, the nonlinear filtering, the intermediate key-expansion RNG's, and the CRC's used to process the User Key. Any of these changes will essentially produce a different cipher.

In our designs we typically use Dynamic Substitution combiners, which are one of our major technologies because they add strength and prevent *known-plaintext* attack. We can vary the number of combiner levels, the number of combiners in each level, and the initialization; each variation would be a different cipher.

We typically create a random *message key* for every message, and then encipher the message itself under the random message key.

Penknife

Penknife is a finished e-mail stream cipher product with many features. Penknife can be customized, or some of its features applied to other ciphers.

In particular, Penknife could be made stronger, especially if the original emphasis on error-resilience could be relaxed.

Cloak2

Cloak2 is a finished strong file stream cipher product with many features. It can also be customized.

One good extension would be to strengthen the message key cipher so that message keys for individual messages could be extracted and disclosed without revealing the message keys in all messages. This would allow an organization to comply with any sort of message disclosure without weakening their overall security.

Dagger

Dagger is a finished fast stream cipher engine which can be customized.

Certainly, most applications do not need the generality built into Dagger. This means that the cipher can be strengthened in most applications.

BLOCK CIPHERS

Block ciphers are mechanisms which emulate a keyed Simple Substitution of a size which is far too large to explicitly describe. Block ciphers process each message in blocks or "chunks," but even if only one byte of a block is needed, the entire block must be ciphered and transported. Ciphering cannot occur until a block is filled.

In an ideal block cipher, each bit of the plaintext should "affect" each bit of the ciphertext, and changing any input bit should change about half of the output bits. This means that all of the uniqueness in each block of plaintext is captured to increase the complexity of the cipher. In this respect, obviously, large blocks are better, but most conventional block ciphers use tiny 8-byte blocks.

Fencing and Mixing Block Ciphers

Technical Terms

- By "fencing" we mean an array of typically byte-width substitution tables. Each 256-byte table corresponds to a keyspace of 1684 bits (provided, of course, that the rest of the cipher will hide the arrangement of the table).
- By "balanced block mixing" we mean a structure which generally takes two input sub-blocks to two output sub-blocks and provably propagates a change in *either* input sub-block to *both* output sub-blocks. (This is sufficient to provide provable avalanche in a block cipher.)

Fenced DES

(init = 31ms, ciphering = 181 KB/sec)

```
                             INPUT
<-------------------------- 256 bits -------------------------->

S S S S S S S S S S S S S S S S S S S S S S S S S S S S S S S S   fencing
-------------------------------x--------------------------------  mixing
---------------x---------------  ---------------x---------------  mixing
------DES------ ------DES------  ------DES------ ------DES------  DES
---------------x---------------  ---------------x---------------  mixing
-------------------------------x--------------------------------  mixing
S S S S S S S S S S S S S S S S S S S S S S S S S S S S S S S S   fencing

<-------------------------- 256 bits -------------------------->
                            OUTPUT
```

Here we have 32 input and 32 output byte-substitutions (S) across a 256-bit block. We also mix (---x---) 128-bit and 64-bit sub-blocks using linear Balanced Block Mixing. Because all data flows through DES, the cipher cannot be weaker than DES. Each substitution or layer is protected by DES and so apparently cannot be exposed without breaking DES and the other substitution layer.

Each 256-byte substitution table is keyed by shuffling under a cryptographic RNG initialized from a User Key. That RNG also produces 4 separate random DES keys. Normally, the keyspace of this sort of cipher is limited by the size of the RNG used to key the substitutions, and it is easy to have a true keyspace of thousands of bits.

The ability to attack the keying RNG depends upon developing the state in one or more of the substitutions, then inferring the RNG sequence. But inferring the RNG sequence can be made difficult or impossible by double-shuffling each substitution. It is not at all clear how an attacker could develop the correct state of any substitution in the first place. Even a single bit error in any input table is guaranteed to produce avalanche, so the extent of solution of these tables cannot be distinguished. Fenced DES was described on sci.crypt in early 1994.

Fenced Tree

(init = 45ms, ciphering = 113 KB/sec)

```
S S S S S S S S S S S S S S S S S S S S S S S S S S S S S S S S   fencing
-------------------------------x--------------------------------  mixing
---------------x---------------  ---------------x---------------  mixing
-------x------- -------x-------  -------x------- -------x-------  mixing
---x--- ---x--- ---x--- ---x---  ---x--- ---x--- ---x--- ---x---  mixing
-x- -x- -x- -x- -x- -x- -x- -x-  -x- -x- -x- -x- -x- -x- -x- -x-  mixing
S S S S S S S S S S S S S S S S S S S S S S S S S S S S S S S S   fencing
-x- -x- -x- -x- -x- -x- -x- -x-  -x- -x- -x- -x- -x- -x- -x- -x-  mixing
---x--- ---x--- ---x--- ---x---  ---x--- ---x--- ---x--- ---x---  mixing
-------x------- -------x-------  -------x------- -------x-------  mixing
---------------x---------------  ---------------x---------------  mixing
-------------------------------x--------------------------------  mixing
S S S S S S S S S S S S S S S S S S S S S S S S S S S S S S S S   fencing
```

This is similar to Fenced DES, but with three more Balanced Block Mixing layers (for a total of five) and another fencing layer instead of the DES layer.

Fenced Quad

(init = 45ms, ciphering = 398 KB/sec)

```
S S S S S S S S S S S S S S S S S S S S S S S S S S S S S S S S   fencing
===============================================================   lin-FFT
======= ======= ======= ======= ======= ======= ======= =======   lin-FFT
S S S S S S S S S S S S S S S S S S S S S S S S S S S S S S S S   fencing
======= ======= ======= ======= ======= ======= ======= =======   lin-FFT
===============================================================   lin-FFT
S S S S S S S S S S S S S S S S S S S S S S S S S S S S S S S S   fencing
```

Here we have two full linear "FFT" operations between three fencing layers. Note that many different mixing arrangements will produce equally acceptable mixing.

Each "FFT" "butterfly" operation is a Balanced Block Mixer of appropriate width. Basically, this design assumes that mixing with 32-bit sub-blocks is likely to be faster than mixing with 8-bit sub-blocks. Consequently, the mixing is broken down into 32-bit "FFT" layers and 8-bit "FFT" layers.

Fenced FFT

(init = 45ms, ciphering = 413 KB/sec)

```
S S S S S S S S S S S S S S S S S S S S S S S S S S S S S S S S   fencing
===============================================================   lin-FFT
S S S S S S S S S S S S S S S S S S S S S S S S S S S S S S S S   fencing
===============================================================   lin-FFT
S S S S S S S S S S S S S S S S S S S S S S S S S S S S S S S S   fencing
```

Here we also have linear "FFT-like" mixing between three fencing layers. But here, every "FFT" "butterfly" operation is a byte width Balanced Block Mixer.

Fenced OLS

(init = 76ms, ciphering = 372 KB/sec)

```
                            INPUT
<------------------------- 256 bits -------------------------->

===============================================================   nl-FFT

<------------------------- 256 bits -------------------------->
                            OUTPUT
```

Here we have non-linear FFT-style mixing, and no fencing layers at all. Each "FFT" "butterfly" is a byte-width Balanced Block Mixer composed of a keyed pair of orthogonal Latin squares of order 256.

This prototype uses a single 128K randomized table, with a keyspace of 3368 bits. But only memory space and initialization time prevents us from using a different table at every mixing node, thus producing an astronomical keyspace.

Variable Size Block Ciphers

The usual block cipher has a small fixed width (often 64 bits). But if plaintext really does contain uniqueness at a rate of about one bit per character, the usual block cipher covers only about eight bits of uniqueness, a searchable quantity. (This sort of attack, of course, would require a deep knowledge of language structure, but doing "Triple" anything would not affect it.) By increasing block size dramatically, we can have a cipher which is not vulnerable to this sort of attack.

The usual stream cipher can cipher sections of arbitrary size, but can only propagate uniqueness to *later* ciphertext. In contrast, a block cipher -- including all of these VSBC designs -- will propagate plaintext uniqueness to *all* bits of ciphertext in the same block.

The Variable Size Block Cipher (VSBC) concept is a new paradigm, and can be used in ways far different than fixed size block ciphers. For example, in a VSBC there is no need to use fixed-size blocks: A message can be ciphered in blocks of *pseudo-random* size. This means that an Opponent cannot even know what data belongs to a single block.

Here I show the structure for typically three adjacent bytes, but each of these designs is used in a file cipher which dynamically selects block size to the byte with an average size of about 2 KB.

These VSBC designs have a very similar structure for both enciphering and deciphering. Data flows down for enciphering, and up for deciphering. The tables (or Latin square rows) used for deciphering are the inverse of those used for enciphering. In the SBBM and OLs designs, the diffusion direction also changes.

In the two designs which use substitution tables, the tables are numbered to remind us that each is generally different. Nor is the table for any particular position fixed: Tables are used as needed from an array of 64 such tables. Each table is separately shuffled by the cryptographic RNG and each table represents a potential 1648-bit key. Normally, the keyspace of this sort of cipher is limited by the size of the RNG used to key the tables or squares, and it is easy to have an efficient true keyspace of many thousands of bits.

In marked contrast to other cipher designs, additional confusion and diffusion layers are easily and modularly added to produce a cipher of any desired working strength.

Performance measurements occurred for RAM-drive ciphering of a 750K file on a P90 under DOS 6.22, with single-pass shuffles.

VSBC Substitution / XOR (SubX)

(init = 15ms, ciphering = 461 KB/sec)

```
               INPUT or OUTPUT
       d[0]        d[1]        d[2] . . .
        |           |           |
iv0 -> XOR  +----> XOR  +----> XOR
        |   |       |   |       |
       S00  |      S10  |      S20
        *-- | -+    *-- | -+    *--> c1
iv1 -> XOR  |  +-> XOR  |  +-> XOR
        *---*       *---*       *--> c0
iv2 -> XOR  |  +-> XOR  |  +-> XOR
        *-- | -+    *-- | -+    *--> c2
       S01  |      S11  |      S21
        |   |       |   |       |
iv0 -> XOR  +----> XOR  +----> XOR
        |           |           |
       XOR <----+  XOR <----+  XOR <- iv4
        |       |   |       |   |
       S03      |  S13      |  S13
  c5 <--*    +- | --*    +- | --*
       XOR <-+  |  XOR <-+  |  XOR <- iv5
  c4 <--*       *---*       *---*
       XOR <-+  |  XOR <-+  |  XOR <- iv6
  c6 <--*    +- | --*    +- | --*
       S04      |  S14      |  S14
        |       |   |       |   |
       XOR <----+  XOR <----+  XOR <- iv4
        |           |           |
       d[0]        d[1]        d[2] . . .
         OUTPUT or INPUT
```

The SubX VSBC design uses exclusive-OR combining in diffusion layers, and keyed, byte-wide 256-entry substitution tables in confusion layers. The particular table used in each position is the "next" table in sequence from an array of keyed tables. (Even if two blocks do end up having the same size, they probably will not have the same tables in the same positions.)

The SubX design uses the most primitive components, and so is graphically more complex than the other designs.

VSBC Latin Square (Ls4)

(init = 55ms, ciphering = 236 KB/sec)

```
       d[0]        d[1]        d[2] . . .
        |           |           |
iv0 -> Lsc  +----> Lsc  +----> Lsc
        *-- | -+    *-- | -+    *--> c1
iv1 -> Lsc  |  +-> Lsc  |  +-> Lsc
        *---*       *---*       *--> c0
iv2 -> Lsc  |  +-> Lsc  |  +-> Lsc
        *-- | -+    *-- | -+    *--> c2
iv0 -> Lsc  +----> Lsc  +----> Lsc
        |           |           |
       Lsc <----+  Lsc <----+  Lsc <- iv4
  c5 <--*    +- | --*    +- | --*
       Lsc <-+  |  Lsc <-+  |  Lsc <- iv5
  c4 <--*       *---*       *---*
       Lsc <-+  |  Lsc <-+  |  Lsc <- iv6
  c6 <--*    +- | --*    +- | --*
       Lsc <----+  Lsc <----+  Lsc <- iv4
        |           |           |
       d[0]        d[1]        d[2] . . .
```

The Ls4 (four Latin squares per data element per diffusion direction) VSBC design uses Latin square combining, which simultaneously provides both diffusion and confusion in the same layer. A single keyed Latin square of order 256 is used, requiring 64K of store and representing a 3296-bit key. (The prototype RNG is a jitterized Additive RNG with 496 bits of initial state.)

The overall structure is quite like the SubX design, but differs in that the "table" (actually, the Ls row) used at each node is here selected by diffusion *data*, instead of some value related to node position. Since The Opponent does not see the diffusion data, it is going to be tough to isolate a particular "table" to be attacked separately.

VSBC Latin Square / XOR (LsX)

(init = 55ms, ciphering = 372 KB/sec)

```
       d[0]           d[1]           d[2] . . .
        |              |              |
iv0 -> XOR  +-------> XOR  +-------> XOR
        *-- | ----+    *-- | ----+    *--> c2
iv1 -> Lsc  |  +----> Lsc  |  +----> Lsc
        *---*  |  |    *---*  |  |    *--> c0
iv2 -> Lsc  |  |  +-> Lsc  |  |  +-> Lsc
        *-- | -+       *-- | -+       *--> c1
iv0 -> XOR  +-------> XOR  +-------> XOR
        |              |              |
       XOR <-------+  XOR <-------+  XOR <- iv4
  c6 <--*     +---- | --*     +---- | --*
       Lsc <----+  |  Lsc <----+  |  Lsc <- iv5
  c4 <--*    |  |  *---*    |  |  *---*
       Lsc <-+  |  |  Lsc <-+  |  |  Lsc <- iv6
  c5 <--*       +- | --*       +- | --*
       XOR <-------+  XOR <-------+  XOR <- iv4
        |              |              |
       d[0]           d[1]           d[2] . . .
```

The LsX VSBC design uses Latin square layers for both diffusion and confusion, and exclusive-OR combining for other diffusion layers. This design also demonstrates a somewhat different feedback architecture.

VSBC Substitution / Balanced Block Mixing (SBBM)

(init = 15ms, ciphering = 439 KB/sec)

```
d[0]    d[1]    d[2]    d[3] . . .
 |       |       |       |
S00     S10     S20     S30
 |       |       |       |
 +----> BBM --> BBM --> BBM -->      -----+
         |       |       |                |
        S01     S11     S21              Sy1
         |       |       |                |
         +----> BBM --> BBM -->      --> BBM -----+
                 |       |                |       |
                S02     S12              Sx2     Sy2
                 |       |                |       |
         +----- BBM <-- BBM <--      <-- BBM <----+
         |       |       |                |
        S03     S13     S23              Sy3
         |       |       |                |
 +----- BBM <-- BBM <-- BBM <--      <----+
 |       |       |       |
S04     S14     S24     S34
 |       |       |       |
d[0]    d[1]    d[2]    d[3] . . .
```

The SBBM VSBC design uses the fast and simple Balanced Block Mixing component in the diffusion layers, and byte-wide keyed substitution tables in the confusion layers. Again, these tables are used in cyclic sequence from an array of keyed tables, so the same tables may or may not occur in the same positions in some other block.

VSBC Orthogonal Latin Square (OLs)

(init = 76ms, ciphering = 369 KB/sec)

```
d[0]    d[1]    d[2]    d[3] . . .
 |       |       |       |
 +----> OLs --> OLs --> OLs -->      -----+
         |       |       |                |
         +----> OLs --> OLs -->      --> OLs -----+
                 |       |                |       |
         +----- OLs <-- OLs <--      <-- OLs <----+
         |       |       |                |
 +----- OLs <-- OLs <-- OLs <--      <----+
 |       |       |       |
d[0]    d[1]    d[2]    d[3] . . .
```

This OLs VSBC design uses a keyed orthogonal pair of Latin squares which realize a keyed Balanced Block Mixer at each node. The OLs structure fills 128K and also represents 3296 bits of key.

SUMMARY

Any of these ciphers or prototypes can be the basis of a new custom cipher.

Terry Ritter, his current address, and his top page.

*Last updated:*1996-02-25

Why Use Our Ciphers?

Terry Ritter

A widely used cipher may ALREADY be broken.

One alternative is to use A WIDE VARIETY of new and different ciphers, each of which requires new effort to break.

Contents

- Introduction
- Why Not Just Use "Proven" Ciphers?
  - There Are None
- Ciphering is Not Just Another Algorithm
  - It May Be a Mistake to Call Ciphering Designs "Algorithms," Ciphering is Only Part of the Problem, Designs with Strong Ciphering Can Still be Weak
- Old Ciphers and Bold Ciphers
  - Many Old Ciphers, No Practical Cipher Has Proven Strength, DES, Triple DES, IDEA, RSA, Others
- Faster Operation
  - Better User Satisfaction, Better for Heavy LAN Loading, Better for Multiple Ciphering
- Greater Strength
  - Cloak2 has a 992-bit keyspace, 4x Fenced DES has at least a 224-bit keyspace, Variable Size Block Ciphers can have a huge keyspace, DES only has a 56-bit keyspace.
- Larger Blocks
  - 4x Fenced DES has a block size of 256 bits, Variable Size Block Ciphers can have a huge block size, DES only has a block size of 64 bits.
- Better Architectures
  - Modern stream ciphers, Block ciphers of larger size and keyspace, Block ciphers of dynamically-variable size, DES has a fixed block size of 64 bits.
- Many Ciphers
  - Having one cipher is putting all one's eggs in one basket.
- VLSI Orientation
  - Regular structures using storage-oriented components.

Introduction

Q1: What problem is addressed?
A1: The problem of storing or transporting sensitive information so that the information is not stolen or copied.

Q2: What is the product?
A2: We have several cipher programs for DOS which also operate well under Microsoft Windows. We also own proprietary technology which allows us to provide better custom designs.

Q3: How does the product solve the problem?
A3: Sensitive information is "encrypted" or "enciphered" before storage or transport. Anyone looking at the resulting file sees only jumbled data.

Q4: But what if someone steals the ciphering program?
A4: A cipher program can be considered a "lock," a user has a "key" which is used in that lock. Stealing the program is not much help unless the key is also stolen. Keys are kept secure, just like house or car keys.

Q5: For our personal use, why would we not use PGP and public keys?
A5: PGP is oriented toward technical individuals who understand the serious consequences of not authenticating a public key.
On the other hand, secret keys can be delivered through normal business channels and allocated to users like any other resource. Our secret-key products do not require a deep technical understanding of cryptography beyond the analogy to metal keys which all of us use to protect our homes and cars.

Q6: For our commercial programs, why would we not use some free design or something taken from a cryptography text?
A6: First, there are many different ciphers, and various tradeoffs between them. Next, there is normally a great deal more involved in the overall use of a cipher than just the cipher design itself. Last, while it is easy to build a toy cipher, surprisingly, most easy ciphers are breathtakingly weak. It generally takes years of study and experience to build good ciphers or even use old ones securely. It seems reasonable to hire this experience rather than trying to reproduce it.

Q7: If cipher design is so tricky, why would we consider using a non-standard cipher?
A7: First, the only real standard is the US Data Encryption Standard, which is generally slow and can be weak if used naively. Nowadays many use "Triple-DES" which is only a third as fast as DES itself. Next, there is no need to standardize on a single cipher design. Indeed, using different ciphers is less dangerous than "keeping all your eggs in a single basket," because no cipher is proven "strong."

Why Not Just Use "Proven" Ciphers?

First, there ***are*** no practical ciphers which are *proven* to be strong! There *are* ciphers which have been around for a while, and which no academic has reported breaking, but those ciphers may well have been broken by other groups who are not talking. Older ciphers may be essentially obsolete anyway, due to obvious increases in computation which attackers now have available.

Next, a particular application may provide reason to select a cipher with special characteristics. Our custom designs can be especially fast, have a particular block size, or support parallel ciphering, among other characteristics. Our component-based designs accommodate special applications; other design technologies are more limited.

Alternately, the ability to handle a wide variety of different ciphers can provide a form of probable security even though none of the ciphers is itself proven "strong." This requires a large number of serious ciphers, which is only likely using some form of component technology like ours.

Last, but not least, cryptographic design involves **more** than simply using a "strong" algorithm. There is more to be done, and if it is not done right, even a *good* algorithm can produce a **weak** cipher system. We support our ciphers, and offer custom design services to complete the system.

Ciphering is Not Just Another Algorithm

It is probably a mistake to think of ciphering designs as "algorithms." Programmers implement "algorithms" all the time, leading to the idea that ciphering is something which any programmer can read about and implement.

But when a programmer implements, say, a division algorithm, many years of schooling provide a good background on how to *test* the program in very substantial ways. When a programmer implements a *cipher*, normally all she can test is that it converts text into a mis-mash and then deciphers that into the original text. Even toy ciphers get those tests right.

When one implements a division algorithm, that solves **the entire problem** of dividing. But when one implements a ciphering "algorithm," this is only **part** of the job of a strong cipher. Other parts include key generation, key distribution or transport, transported key certification or validation, and a whole host of other things.

If a public key cipher is used for distributing keys over the Internet *without* key certification, that cipher must be considered explicitly "weak." Even *if* all of the "algorithms" in the design were in fact "strong," *and* all were implemented properly, the result would *still* be "known to be weak."

The simple *use* of "strong" algorithms *does not* guarantee that a cipher will not be "weak."

Cipher "algorithms" solve only *part* of the requirements of a real cipher system.

Old Ciphers and Bold Ciphers

There are many *old* ciphers, and there are many ciphers which are claimed to be "strong." But there are **no** practical ciphers, old *or* bold, which are *proven* "strong."

Any widely-used cipher is precisely the sort of thing which someone might wish to attack, and, having been successful, might wish everyone else to use.

Since attackers will not tell us when they are successful, the only reasonable alternative seems to be to use a wide variety of different ciphers, thus forcing each of these to be attacked separately, with far less payoff.

DES

The U.S. Data Encryption Standard (DES) was standardized in 1977, and designed before the microprocessor revolution. It uses a small amount of storage for internal tables, which limits the amount of nonlinearity or uncertainty in the internal structure. Such tables as there are are not keyed.

While DES *seems* to have withstood attack, we actually do not know if this is true. If some group *had* found how to break DES, it would be in their interest not to reveal this, so that the group could profit from their work.

We do know that DES is relatively slow, with a fixed and awkward architecture. Moreover, the small 56-bit DES keyspace has meant that new attack technology (for example, a $1,000,000 system made from custom chips) has made DES vulnerable to the most obvious attack (brute force on the keys). DES also has a small 64-bit block size which at the very least limits the amount of information which should be sent under a single key.

Newer DES attacks, such as those intended to reveal secret information hidden in "smart cards," while too new to place in their proper context, seem absolutely devastating, and undermine our confidence in the design itself.

The known weakness of DES means that DES alone is probably not a good solution, even if its excessive computation and restrictive block cipher design are acceptable.

Triple DES

The main contender to succeed DES is *Triple DES*, which is just DES done three times with different keys. This supposedly increases the keyspace to about 120 bits, but the same old 64-bit block size remains.

Triple DES needs three times the computation of even DES itself, and is not *guaranteed* stronger than DES. In fact, the recent "smart card" attacks break Triple DES almost as easily as DES.

IDEA

"IDEA" (also called PES and IPES) is a European design perhaps best known for its use the "Pretty Good Privacy" (PGP) program.

While IDEA has a large 128-bit keyspace, it still has the usual small 64-bit block size. IDEA also uses internal operations which are *almost* linear, and this is a little disturbing.

RSA

The Rivest-Shamir-Adleman (RSA) algorithm was the first practical public-key algorithm. While there are other public-key algorithms, none is clearly better or faster, except under particular conditions.

The RSA computations involve huge numerical values, and some fairly complex implementation. RSA is normally used simply to deliver a message key which is then used by one of the "conventional" or "secret key" ciphers. This of course means that selecting a "secret key" cipher is still an issue even in "public key" ciphers.

Because RSA computations involve huge numerical values, they require keys perhaps 10x as long as secret key ciphers. If we like 120-bit secret keys, we should be looking at public keys of at least 1200 bits.

Others

The past few years has seen a vast expansion in the number of publicly-known cipher designs. Clearly it is not possible to address them all, but we *can* make a few observations about "conventional" or "secret key" designs:

- When the keyspace is under, say, 120 bits, we need to wonder how the cipher can survive a decade or two of use with rapidly-increasing attack capabilities.
- When a block cipher has a small size block, we are forced to limit the amount of data ciphered under a single key, and this must be part of the cipher design.
- Ideally, a block cipher would have a size such that we could expect a plaintext block to carry more "uniqueness" or "entropy" than twice the key length. If we like a 120-bit key, this might imply a 240-*byte* block, which is about 30x the usual 64-bit (8-byte) block.
- No cipher can force an ordinary user to remember a long, random key. Therefore, there should be intermediate stage where such a key is looked up in a ciphered file using a simple, public alias.

Obviously, new approaches are subject to new attacks. Since we cannot *prove* strength, we can only continue development on how one *might* attack new designs. But the "alternative" of using old "proven" designs is no alternative at all, since these designs are also not proven, and there is ample motive for those who can break the cipher to mislead us that it is strong.

Faster Operation

Ciphering is an overhead to communication: The less computation required for ciphering, the more available to move data. (Normally the issue is the load on central servers, as opposed to end-user machines, but of course both ends must use the same cipher.) New technology can be both stronger *and* faster than ordinary DES.

High speed buys a lot:

- An overall faster system of course provides better user satisfaction.
- A fast cipher is ideal for general LAN traffic. Most traffic does not need ultimate protection, and servers are often heavily loaded. The relatively few messages which are especially sensitive can be protected with *additional* higher-level and slower cipher(s).
- Two or more layers of fast ciphering can be faster than a single older cipher, and will also avoid putting all one's security eggs in a single cipher basket.

Greater Strength

DES uses a small 56-bit key. But we can easily construct faster ciphers with far larger keys, with ciphering speed generally independent of keyspace:

- The Cloak2 cipher, for example, has a 992-bit keyspace, and uses a new random 992-bit message key for every message.
- 4x Fenced DES has a keyspace of at least 224 bits, yet ciphers much faster than Triple DES.
- Variable Size Block Ciphers can have a huge keyspace, yet still cipher extremely fast.

Larger Blocks

DES has a 64-bit block, and new technology has made this perhaps too small. Certainly a larger block will contain more plaintext, and also more variation in plaintext; this makes the cipher stronger. We can easily construct ciphers with far larger blocks which still cipher faster than DES:

- 4x Fenced DES has a block size of 256 bits, yet uses DES itself as a trusted component. Associated 2x and 1x versions limit data expansion to only that of DES itself.
- Variable Size Block Ciphers can have an essentially unlimited block size: Fast prototypes with (average) 16,000-bit blocks have been tested.

Better Architectures

DES operates on fixed-width 64-bit blocks of data. This means that:

- all data must be re-cast into 64-bit blocks,
- we must fill a block before we can cipher it, and
- the final block will generally be only partially filled but must be wholly ciphered anyway.

Instead of being confined to the old architecture of DES, we can design:

- modern, strong, stream ciphers,
- block ciphers of larger size and keyspace, and
- block ciphers of dynamically variable size.

Each option will provide advantages (and disadvantages) in particular applications.

Many Different Ciphers

If we are going to put all our security eggs in one basket, we had better be pretty sure of that basket. Unfortunately, with ciphers, this is something we simply cannot do: There *is* **no** practical cipher which is proven strong. So it only makes sense to place some eggs in each of several different baskets, or even to *nest* baskets (ciphers) within one another.

Having a wide variety of fundamentally-different ciphers means far more than just adding a few bits to the keyspace of an existing cipher: If the existing cipher is broken this will probably affect all possible keys. But if one of the new ciphers is broken, others will likely remain operational.

It is important to see the many cipher environment as more than four or five different ciphers. We should plan, in 20 years or so, to support *tens of thousands* of ciphers, with many different fundamental approaches. Only in this way can we force an Opponent to make massive investments for minimal payoffs. Only in this way can we enforce a form of probable security, while using ciphers which cannot be proven secure.

VLSI Orientation

DES was designed in a hardware-poor era. The internal DES operations contain only a small amount of nonlinearity, and what nonlinearity there is was fixed for all users over all time.

In contrast, modern cipher designs occur in a hardware-rich era, provided the design uses regular structures and simple interconnections. Modern ciphers can afford to use large tables which are set up by the ciphering key, and so are different for every use and every user.

The regular structures used in tables are the ideal sort of VLSI design, and present a clear contrast to the "random logic" approach of DES as well as many more-modern ciphers. Our designs are generally more VLSI-friendly than most.

Terry Ritter, his current address, and his top page.

Last updated: 1996-11-26

Cryptography is War!

Current e-mail crypto designs are just *asking* for trouble.

Terry Ritter

Only a few different ciphers are used in e-mail crypto products. Some manufacturers talk of someday allowing users to select from six or eight popular ciphers. I say: "Big deal!"

If someone wants to spend time and money attacking a cipher, they are not going to bother with a cipher which protects almost no data. An attacker will select a widely-used cipher, because that has the best payoff. If you *use* a widely-used cipher, you have the best chance of being exposed!

E-mail systems can and should have a common interface and support *arbitrary* external ciphering routines. In a decade or two we might have *tens of thousands* of different custom ciphers, any of which might be used by the *same* e-mail interface. Each of these ciphers will carry only a tiny fraction of the information protected by a widely-used cipher, and so will be much less attractive and much less likely to be attacked and penetrated.

Security Problems

Messages by Ritter to the "resolving-security mailing list."

Follow the thread, and form your own opinion.

Cryptography is *War!*

The past has seen the standardization of a *single* cipher (DES) for most use. The future will see a multitude of *different* ciphers which users can select from and agree to use. This will improve security for everyone by reducing the amount of information under any particular cipher, thus reducing the motive and resources for anyone to attack that cipher.

> Cryptographers try to build secure ciphers, and cryptanalysts try to break those designs. When an academic cryptanalyst breaks a cipher, we just use a new cipher. But when a team of private cryptanalysts break a cipher, why would they announce it? The continued use of a broken cipher is the cryptanalysts' reward, and a resource to exploit.
>
> Cryptographers always *think* they use strong ciphers, but even after 50 years of mathematical cryptology, **no** practical cipher has been **proven** secure. Historically, *many* ciphers *have* been broken, sometimes after they have been in trusted use for many years. Fortunately, there is a practical alternative to proving a cipher secure: Give many different user-groups each their own *custom* cipher and change ciphers periodically. This reduces the value of the information under any *particular* cipher, which reduces the motivation (and resources) for attacking each cipher.
>
> A system with *many* ciphers is **not** the same as a system with *one* cipher and a slightly larger keyspace. The difference is that any widely-used cipher may *already* be broken and insecure, and *we would not know!* Just changing the *key* in this situation probably **will not** recover secure ciphering. But in a many-cipher environment, changing the *cipher* probably **will** recover security: Direct attacks on serious ciphers are expensive, and resource limitations naturally discourage efforts which will be unprofitable even when successful. In a many-cipher environment there is less information of value under any particular cipher, so there is less reason to attack any of them. Compare this to the current situation, where virtually *all* information is "protected" by one of a small handful of widely-used ciphers. Widely-used ciphers make *ideal* targets.
>
> It is sometimes said that a multi-cipher environment is not suitable for "widespread interoperable use," because it is necessary to have the same *cipher* on both ends. But it *always* has been necessary to have the same (or related) *key(s)* at both ends, and doing this is a major part of modern cryptography. Since a key-delivery process could deliver **both** a *key* **and** a *cipher*, custom ciphers are easily supported (provided we have a good cipher interface).

Having many different ciphers is an aggressive and pro-active strategy for winning the cryptography war. Our company has the knowledge, technology and experience to build a wide range of unique custom ciphers. We aggressively prosecute the cryptography war for the security of our clients and customers.

Terry Ritter, his current address, and his top page.

Last updated: 1996-06-20

Our Patent Policy

Terry Ritter

We patent our work and license our patents.

Contents

- Brief Summary
- Background
- Claims Tutorial
- The Law

Brief Summary

Even the barest mention of patents can lead to adversarial relationships which neither exploit nor advance the technology. But personal use is rarely a significant issue, and in corporate relationships we can be a form of ongoing outside research. Certainly we will be most inclined to share future results with our friends and supporters.

Personal Use

We prefer not to know about someone's personal use. But if for some reason you *need* to operate under a specific formal license, you **must** contact us and describe the limits of your proposed use. A positive response from us will contain your limited personal usage license number.

Corporate Use

We offer non-exclusive licenses to our technology. Because of the number of variables involved and various special cases, there really is not much point in stating a particular percentage or fee. In many cases, it will be beneficial to have us construct an appropriate custom but non-exclusive cipher. Often there will be a continuing interest in the field and thus a desire to suggest directions and learn early results which are not likely to be available in an adversarial relationship.

Background

Perhaps because we publish our work, some people are disturbed that we protect our developments by patent. But patents are *normal* in the research business. *Most* R&D (research and development) engineers are *required* to assign their patent rights to their employer. Even *academics* generally cede patent rights to the institutions which pay their salary, and which provide lab space and equipment. Companies and academic institutions often license their patents to offset their costs.

In contrast, we take no public funds, and are employed by no large corporation. We perform our own R&D, on our own time, and our own dime. We patent our results so we can eat, live, and continue the work. We see patents as substantially better for society than keeping developments secret so that only we understand their consequences. We also do not subscribe to the socially corrupting idea that someone should do real work and then release it for free without profit. Profit in the form of *wages* is how most of us manage to live.

Claims Tutorial

Patent law is a specialized area of a specialized field. Many common terms and expressions have special meaning in this particular context. Further, patent claims are interpreted or understood in the context of a particular patent. However, it is normally possible to follow at least some claims.

When we look at patent claims, we notice a distinction between those which reference some earlier claim, and a few claims which have no such reference. Claims which do *not* refer to other claims are called "independent," and carry the broadest coverage because they are constructed to have the fewest requirements and limitations. Claims which *do* refer to other claims are called "dependent" claims, and these *include* the requirements and limitations of some other claim, further narrowed with *additional* requirements and limitations.

The reason for having *dependent* claims is that it is *impossible* to know everything which was published or in common use anywhere in the world. Therefore, a broad claim may well someday be found to have been "anticipated," and ruled unenforceable. In this case we hope that the increasingly fine restrictions of the dependent claims will remain unanticipated and, thus, still in effect. The Patent and Trademark Office (PTO) tries to limit patents to typically 3 independent claims.

Each claim stands alone in defining some particular range of coverage. To find infringement, it is *not* necessary that *every* claim apply to some art, only that *one* claim apply fully. A claim is said to "read on" infringing art when each requirement and limitation in the claim is satisfied. The actual art may have a great deal of elaboration, but infringement only requires that the claimed material be *present*, not *exclusive*.

Here is claim 1 from the Dynamic Substitution patent:

```
1. A mechanism for combining a first data source and a second data
source into result data, including:

     (a) substitution means for translating values from said first
data source into said result data or substitute values, and

     (b) change means, at least responsive to some aspect of said
second data source, for permuting or re-arranging a plurality of the
translations or substitute values within said substitution means,
potentially after every substitution operation.
```

Suppose it is proposed to use something an awful lot like Dynamic Substitution to produce an authentication value for a message:

Here we can take the message as a "data source," the RNG confusion stream as another "data source," and the hash value as the "result data." We clearly have a "substitution means" or table. We also have a "change means" which makes changes in the table, and said "change means" is indeed responsive to the RNG confusion stream which controls the shuffling. That's it. Claim 1 reads on the proposed structure.

The Law on Patent Infringement

Patent law is contained in Title 35 of the US Code, which is available on the net. The most interesting part of the chapter on infringement is this:

```
UNITED STATES CODE
          TITLE 35 - PATENTS
               PART III - PATENTS AND PROTECTION OF PATENT RIGHTS
                    CHAPTER 28 - INFRINGEMENT OF PATENTS

§ 271. Infringement of patent
     (a) Except as otherwise provided in this title, whoever without
authority makes, uses or sells any patented invention, within the
United States during the term of the patent therefor, infringes the
patent.
     (b) Whoever actively induces infringement of a patent shall be
liable as an infringer.
     (c) Whoever sells a component of a patented machine, manufacture,
combination or composition, or a material or apparatus for use in
practicing a patented process, constituting a material part of the
invention, knowing the same to be especially made or especially
adapted for use in an infringement of such patent, and not a staple
article or commodity of commerce suitable for substantial
noninfringing use, shall be liable as a contributory infringer.
```

Note that the term "sells" in paragraphs (a) and (c) is generally taken to include free distribution.

While there are always exceptions, normally it would not make sense to take individual users to court. Nor does it make sense to price a manufacturer out of the market. A patent is best seen as an *economic tool*, rather than an end in itself.

Terry Ritter, his current address, and his top page.

Last updated: 1997-03-31

The DAGGER API and Usage

An Easy-To-Use Cipher Engine

Terry Ritter

DAGGER is a set of ciphering routines which add serious information security to a larger overall program. DAGGER is available in "portable C" and 80x86 assembly language.

To use DAGGER, the programmer must first allocate a 13K structure called a DRT (Dagger Record Type). The programmer then calls the DAGGER SetUp routine which initializes the DRT from a key.

After SetUp, the programmer calls the DAGGER Encipher and Decipher routines which each use a DRT to cipher an arbitrary-size memory buffer "in place." That's it!

DAGGER can be used for LAN data, disk ciphering, or other applications. With multiple DRT's, DAGGER can easily interleave operations on different keys.

Note that DAGGER is not appropriate for every possible ciphering situation. In particular, DAGGER is vulnerable to a substantial *defined-plaintext attack* and should not be used in ways which support such attack. DAGGER is appropriate for local storage protection, and for end-to-end communications ciphering. DAGGER is less appropriate for network hops which carry messages from many users over a long time without changing keys.

- Introduction
  - How is DAGGER Different?
- The API
  - The DRT Structure, The Set Up Function, The Encipher Function, The Decipher Function, DRT Structure Details
- Portable C
  - Integer Size, Byte Ordering, Storage within C Structures Not Necessarily Contiguous, Optimization
- 80x86 Assembly Language
  - Machine Addressing, Extra Register Storage, High Level Interface
- Usage
  - Multiple Key's Mean Multiple DRT's, Encipher/Decipher in a Single DRT, Good Keys are Important, Key Distribution, In-The-Field Automatic Strength Upgrade, We Can Avoid Restart (Given Sequential Ciphering Without Errors), We Can Use Message Keys (If Data Expansion is Acceptable)

Introduction to DAGGER

DAGGER is a fast, reasonably-secure software cipher. Because all software processing takes time, there is inevitably a tradeoff between strength and speed in a software cipher. In DAGGER we have an especially good tradeoff, because we can use our own patented Dynamic Substitution technology.

As with any good cipher, the easiest ways to expose DAGGER-protected information probably would not include trying to "break" the cipher. Instead, it is important to consider how plaintext documents are developed, kept, displayed and disposed, who has access to secure data storage, and whether or not workers can be bribed into revealing information. There are probably many other avenues to any protected information which would be far simpler than trying to "break" the cipher.

How is DAGGER Different?

DAGGER is easy to use! In particular, DAGGER is designed to not impose upon a surrounding system. Existing software systems generally do not have to be re-designed to use DAGGER. Thus, DAGGER holds out the novel promise of "drop in" data security.

DAGGER is extremely planned in accepting virtually any set of key, and any amount of data for ciphering. DAGGER ciphers the data "in place," and never needs even one additional byte of storage for the result. DAGGER allows data "frames" to be deciphered in random order, and yet still protects against "known plaintext" attacks.

The API

The DAGGER API consists of one data structure type called DRT and exactly three routines: **SetUp**, **Encipher** and **Decipher**. That's it!

The DRT Structure

The DRT (Dagger Record Type) structure takes about 13KB and holds the various tables and values used in operation. It is allocated by the programmer. In most cases, the programmer will not need to understand the DRT structure at all, and will just allocate storage for it. However, the structure is discussed in detail at the end of this section.

The SetUp Function

The SetUp call initializes a DRT structure from a User Key. The User Key may be any contiguous sequence of bytes; it could be an ASCII text phrase or a sequence of random values. The C prototype looks like this:

```
void
DaggerSetUp( DRT *DR,
             BYTE key[], WORD keybyct, BYTE StrengthBytes );
```

Mainly, DaggerSetUp takes a pointer to a DRT structure, and a pointer to a contiguous key, plus a key length. Then it fills the DRT.

The StrengthBytes value limits the strength of the DAGGER cipher; a value of 5 would restrict the cipher to 40 bits of randomized key. When DAGGER is used internationally, using a StrengthBytes value higher than allowed by regulation could have serious repercussions.

The Encipher Function

The encipher call uses an initialized DRT to encipher a buffer. The buffer may be any contiguous sequence of bytes, and the buffer is ciphered in place: cipher bytes replace the original bytes. Here is the C prototype:

```
void
DaggerEnc( BYTE DatBuf[], WORD byct, DRT *DR, BOOL Restart );
```

When TRUE, the Restart value causes the cipher to "start over"; this is the normal situation when data frames may be deciphered out of sequence or at random. By starting over each time, each frame is enciphered independently.

When the Restart value is FALSE, the cipher does not "start over," but simply continues from the previous enciphering. This could be used for enciphering whole files, when the data are known to have a fixed sequence. It could not be used for low-level LAN ciphering, because error-correction may cause a frame to be re-sent out of sequence.

The Decipher Function

The decipher call looks just like encipher, and uses an initialized DRT to decipher a buffer. The buffer may be any contiguous sequence of bytes, and the buffer is ciphered in place: cipher bytes replace the original bytes. Here is the C prototype:

```
void
DaggerDrDec( BYTE DatBuf[], WORD byct, DRT *DR, BOOL Restart );
```

When TRUE, the Restart value causes the cipher to "start over"; this is the normal situation when data frames may be deciphered out of sequence or at random. Any data errors in the ciphertext which occur during transmission or storage will be confined to a single data frame.

When the Restart value is FALSE, the cipher does not "start over," but simply continues from the previous deciphering. Any data-transmission errors which occur will be compounded and expanded in subsequent data frames.

DRT Structure Details

Here are the DRT (Dagger Record Type) definitions:

```
#define SUBBOXLAST  255
#define DRTPRELAST  1023
#define DRTRANDLAST 4095
#define DRTSHFLAST 1023

typedef struct {
        BYTE  FwdOrg[ SUBBOXLAST+1 ], FwdInse[ SUBBOXLAST+1 ];
        BYTE  InvOrg[ SUBBOXLAST+1 ], InvInse[ SUBBOXLAST+1 ];
        WORD  PreAR[ PRELAST+1 ];
        WORD  RandAR[ RANDLAST+1 ];
        WORD  ShufAR[ SHUFLAST+1 ];
        WORD  RNGencCur, RNGdecCur, RNGorg, Dummy;
} DRT;
```

SUBBOXLAST defines the last element in a 256 element "SUBBOX" array. This is a "substitution box," an array dedicated to substitution transformations. The values in the array are some permutation of the index values 0..255.

DRTPRELAST defines the last element in the "PRE" area of the DRT structure.

DRTRANDLAST defines the last element in the "RAND" area of the DRT structure.

DRTSHUFLAST defines the last element in the "SHUF" area of the DRT structure.

FwdOrg is a subbox containing the initial Dynamic Substitution state. This state is created by shuffling a "standard" ordering (0,1,2,...,255) using the values from the SHUF area. The FwdOrg state is retained unchanged so that the cipher may later "restart" in exactly the same state. The FwdOrg area is originally used to hold the hashed and strength-limited key values during key expansion; initializing FwdOrg automatically destroys those values.

FwdInse is a subbox which holds the dynamically-changing cipher state. FwdInse is initialized by copying from FwdOrg on each restart.

InvOrg is a subbox containing the inverse of FwdOrg. It is also retained unchanged.

InvInse is a subbox which holds the dynamically-changing inverse cipher state. InvInse is initialized by copying from InvOrg on each restart.

PreAR is a 2KB (1k word) array which provides the necessary room to build up from deg 9 and deg 89 to the deg 1279 expansion polynomial.

RandAR is a 8KB (4k word) array is used to store the "random" values used by the cipher. All values in this area were originally created by the deg 1279 expansion polynomial and were subsequently non-linearized by the jitterize mechanism.

ShufAR is a 2KB (1k word) array which provides room for the jitterize mechanism to delete about 10% of the traversed data, and yet leave the RandAR area fully processed. In addition, the values at the start of ShufAR are used to shuffle the original forward substitution box which is the initial Dynamic Substitution state. The entire process of expansion, nonlinearization, and shuffling are necessary in order to generate the cipher state from a key.

RNGencCur is the current encipher index into RandAR, and is set from RNGorg on Restart. If set externally, RNGencCur must be masked by an AND with RANDARLAST to keep the value inside RandAR.

RNGdecCur is the current decipher index into RandAR, and is set from RNGorg on Restart. If set externally, RNGdecCur must be masked by an AND with RANDARLAST to keep the value inside RandAR. Having separate encipher and decipher indexes allows the same DRT structure to be used both for enciphering and deciphering, which is useful for data storage ciphering.

RNGorg is the starting value for the RandAR indexes. If set externally, RNGorg must be masked by an AND with RANDARLAST to keep the value inside RandAR. RNGencCur and RNGdecCur are copied from RNGorg on Restart. RNGorg could be used as a small but efficient message key, for message key values somehow made available by the rest of the system.

Dummy is used to round out the structure to an even length in four-byte elements.

Portable C

The portable C version is intended to function the same -- that is, encipher the same data in the same way, given the same key -- on any machine having a C compiler. These good intentions may be almost impossible to meet fully, but the portable version should do fairly well.

Integer Size

The usual problems with different size integers on different machines are localized by defining appropriate storage types. In this case, BYTE, WORD and LWD are intended to represent 8-bit, 16-bit, and 32-bit unsigned values respectively; these can be re-defined for particular machines and compilers as appropriate. In general, larger storage elements could be used, except that we expect to be able to index the User Key, data, and substitution tables as BYTE arrays and the first two of these are likely to be defined outside the cipher system. All computations are allowed to overflow. No signed values are needed or used.

Byte Ordering

Another common portability problem is byte-ordering, and this might seem to be a particular problem for DAGGER, because the first phase of the set-up algorithm fills the DRT structure with WORD values, and a subsequent phase uses BYTE values from the DRT to permute a translation table. However, the BYTE interpretation of this storage is made portable by reading only WORD values, and then explicitly selecting the high or low BYTE of those values using the least-significant-bit of a count value.

The second phase of setup creates the BYTE-array translation tables on top of the WORD-array storage used during key extension. This also is not a byte-ordering problem because the BYTE tables are not set up until after that area ceases to be used for WORD access, and they are not used for BYTE access until after they have initialized as arrays of BYTE values.

It might be possible to define the DRT structure as a union, but since the first phase of set up simply uses the structure as a contiguous WORD array, it is hard to imagine how a union could possibly be clearer than simply casting the structure pointer to a WORD pointer.

Storage within C Structures Not Necessarily Contiguous

C structure elements are not guaranteed to be contiguous, which theoretically might result in some implementations having unused storage between structure elements. This could be a problem because it would appear as extra storage when considered as an array of words. This would result in a shifted sequence in the RNG table, which would certainly cipher differently. Defining a union would not change the problem.

However, since the early and intermediate structures all have a power-of-two size larger than any reasonable register, it is unlikely that any C implementation would waste storage between them. The few lone WORD values which reside in the structure are carefully placed at the far end, where any extra storage should have no effect on the algorithm.

Optimization

Because the Portable C version is "portable," it is also non-optimal. The portable version can be modified to optimize a particular environment; while the resulting version may not be portable, that may not matter. The normal way to build efficient versions on new architectures is first to get the portable version working under a small driver. Then make small changes in (a copy of) the portable program, each time measuring speed and checking that it will encipher to and decipher from the original version.

It is difficult to say what optimization would be most important, because that depends upon the normal usage of the system. At first glance one might think to run through the set-up process and especially reduce the inefficient portable access to byte values within words, but if set-up is infrequent, making SetUp efficient is almost irrelevant.

It will generally be worthwhile to improve the ciphering inner loops as much as possible. But these loops are already fairly tight C, so it is not clear how much more can be done. For example, the data-array access is already done by moving pointer, instead of individual indexings, and the Dynamic Substitution appears to require explicit indexings, and so cannot be improved the same way. Optimizing the BYTE and WORD access for a particular architecture should yield something, but these are only used once each in the inner loops. Other than using a smart compiler, it is not clear that much more can be done at C level (I would love to hear about any improvements which may be found). Assembly-language could help a lot, of course.

One possible speed optimization in C would be to introduce new local pointers to the DRT elements used in the inner loops. It may well be that some compilers will have trouble recognizing the indexing sequence as a single instruction, and by pre-computing the start of each element, simple and fast indexing may be possible even on less-than-optimal compilers.

It would be possible to make separate ciphering routines for each statically-allocated structure, thus allowing direct memory access to the structure instead of indirect access through a pointer. This might (conceivably) save up to 30% in execution time, but would also be dangerously inflexible.

80x86 Assembly Language

The assembly-language version uses the same interface as the portable C version, but is considerably faster. Upon assembly, the resulting .OBJ file can be imported into C for use.

The 80x86 assembly-language version is 16-bit code, generally optimized for 80486 operation. Surprisingly, most 486 optimizations remain compatible with the 8088. The 486 optimizations simply tend to use different instruction sequences than one would have preferred to use on the earlier processors.

The same code is easily expanded to use 32-bit addresses and pointers, and should require little more than using expanded register names. The cipher algorithm proper does not need, and is not particularly helped by, 32-bit operations. There would be some advantage in being able to use 32-bit pointer offsets, thus possibly avoiding the need to save and change segments, and avoiding also the need for a segment-override for data access. Speed might increase somewhat.

Machine Addressing

Intel base-plus-constant offset-plus-register offset addressing fits this application well. In general, we have a pointer to a DRT structure (base), then must select a particular array from the DRT (constant offset), and index into that array (register offset). This can all be done in one instruction, and, on the 486, that instruction takes just two cycles. But this code will function even on a lowly 8088.

It could be a little faster to pre-compute the starting addresses of the necessary tables and save each address in a local. This would support single-cycle 486 access (and would still be 8088 compatible), provided that the necessary addresses were already in the registers. But the pre-compute and each local access to get the addresses into registers would be overhead.

Extra Register Storage

Limited register storage is a perennial problem for assembly-language programming. Because the 486 has 32-bit registers and the algorithm needs only 16-bit words, we might seek to use the hidden higher word for temporary storage. The 486 has a BSWAP instruction which will rotate the current word into the higher word and vice versa.

Unfortunately, even the one cycle BSWAP instructions quickly become significant overhead in an inner loop, and it is confusing to use them this way. In the end, it was possible to avoid using BSWAP in return for a single memory decrement of a local per loop (for decipher only), at a presumed extra cost of one 486 cycle. This also retained 8088 compatibility.

High Level Interface

The classic style of assembly-language programming passes values in registers. While this seems both simple and obvious, it turns out to be not quite as simple as it looks, since the passed values may have to be stored on the stack anyway, to free up register resources for use in the target routine. Ideally, each called routine would protect (that is, stack) all used registers, but this would be almost as expensive as a full-blown HLL procedure-call protocol.

It is just barely possible to pass values in registers with the final DAGGER API (although some internal routines could have problems). But passing values in registers is awkward for calling assembly-language from a high-level-language, since there would have to be some intermediate routine to grab the values from the HLL call stack and set the registers for the assembly-language. Instead of duplicating the interface in this way, it was decided to use the HLL interface at the assembly-language level.

Another issue is the particular HLL interface, the parameter-passage sequence and responsibility for removing parameters from the stack. Frankly, the Pascal conventions are both smaller and faster (if only just a little). However, the assembly-language was set up to use either Pascal or C protocols, and both were linked into a C driver to make sure this would work. It did.

Usage

The DAGGER API is easy to use: First, allocate storage for a *DRT* structure, either static or dynamic. Next, use *SetUp* to process a User Key and set up the DRT structure. Then, use *Encipher* or *Decipher* to cipher data in place, using a particular DRT. That's about it.

Multiple Key's Mean Multiple DRT's

Clearly, the DRT structure holds the "state" of the cipher; multiple DRT's can hold multiple such states. Obviously, the same code can cipher between multiple targets (with a different for each), simply by using different DRT's. Alternately, it would be easy to use two or more DRT structures and multiple-cipher the same data; this could produce the effect of a substantially stronger cipher. Another option would be to convert from one key to another by first deciphering under one DRT and then enciphering under another.

Encipher/Decipher in a Single DRT

Both Encipher and Decipher can share the same DRT, provided they use the same key (of course), and provided that they handle Restart properly. When both ciphering modes share one DRT, they also share a single FwdInse subbox; Restart must be asserted after changing between Encipher and Decipher (to re-init FwdInse). Thus, it would not be possible to share a DRT if enciphering and deciphering operations were intermixed and Restart not used. For most expected applications, Restart will be used on each frame, so there is no problem sharing a DRT. (The alternative is simply to use a separate DRT.)

In data storage applications, just a single key is commonly used, so using a single DRT is just fine. On the other hand, in LAN applications, it is better practice to use a different key in each direction than to try and share a DRT. We want to minimize the amount of data ciphered under any one key, and one way to help do this is to use a separate key each way.

Good Keys are Important

The only unique thing DAGGER really needs is a key. And, since DAGGER processes each key with multiple CRC polynomials, we do not care whether or not the key is ASCII text or random data.

Cryptographically speaking, any key must contain sufficient "uncertainty" to make a "brute force" attack unreasonable. In general, each key should contain 80 bits of "uniqueness" and "uncertainty." **It is impossible to have a short secure key**, and this is true for *any* cipher.

If the key is ASCII text, each one should be at least 25 or 30 characters long and preferably more; each should be a "phrase" instead of a "password." Each ASCII key should contain some numbers and/or special characters, and should have peculiar punctuation, spacing and/or capitalization, to make the phrase different from anything ever printed on paper.

If the key is "random" binary, it should be at least as long as the maximum StrengthBytes value ever to be used. It would help if the key were created by some process which could be considered "really" random. We want to avoid simply issuing keys as sequential values from some random number generator, unless that data is itself first enciphered. We might consider accumulating deviations in LAN usage in a significant amount of CRC state for use as key values.

When ciphering LAN data (or any form of data communications), using the same key for transmissions in both directions is unnecessarily risky. It is recommended that a different key (a different DRT) be used to cipher in each direction

Key Distribution

DAGGER is a "conventional" (or "one-key," or "symmetric," or "secret key") cipher; that is, enciphering and deciphering are both performed with exactly the same key. This is ideal for protecting most forms of stored data, since the original user does not have to give the key to anyone else.

A "secret key" cipher is also ideal for use on a LAN where there is a "key server" and each station has its own secret key to the server node. When two stations wish to communicate, they request a common key from the server, which sends a temporary key to each station, each enciphered in the correct station key.

However, when cryptography is used for data communications, key delivery can be awkward, because each and every key must be somehow delivered to the far end (and retained) in absolute secrecy. On the other hand, this requirement also assures that each end gets exactly the same key, something which simple "public key" systems rarely guarantee.

So-called "public key" (or "two-key," or "asymmetric") technology can simply remove key delivery. But, because public-key ciphering is extremely slow, it is only rarely used to encipher data; instead, public-key technology is normally just used to deliver keys for a conventional cipher. DAGGER could be the conventional cipher.

In-The-Field Automatic Strength Upgrade

As time passes and computers become faster, a "brute force" attack on a limited-size key will become practical for increasing numbers of organizations. Presumably the government will recognize the problem and, at some point, increase the size of the authorized for international ciphering.

In LAN or WAN usage, it is always necessary to transport session keys for the data cipher. We can prepare for the future by normally having larger than necessary keys and also transporting the strength value. The strength value will limit the keys to the acceptable strength level. But when the day comes to upgrade the system, it will only be necessary to have the key server send a larger strength value to have all nodes automatically upgrade their cipher strength. This would be an in-the-field automatic strength upgrade without changing the fielded code at all.

We Can Avoid Restart (Given Sequential Ciphering Without Errors)

During ciphering, the Dynamic Substitution "state" or "permutation" changes; this is what makes this combiner hard to attack. Normally, Restart will be set TRUE, and the system will start over again for each and every data frame. But, if we have an installation which has the right conditions, we can avoid starting over every time. We can avoid doing a Restart on every data frame if:

1. we know that we will always decipher data frames in exactly the same sequence in which they were enciphered, and
2. we are relatively sure there will be no data errors. (A data error in one frame will be retained and will propagate through any remaining data frames until the next Restart.)

The first consideration eliminates storage systems which need random access to data.

The second consideration eliminates most low-level data communications, except those which will detect errors and re-synchronize (and can issue a Restart on that condition). Note that any data frames in transit could be lost unnecessarily in this situation. Normally, a low-level communications protocol will only issue "resend" requests for particular bad frames, and will not restart the communication stream after every single error. An error-correcting modem might do this, however.

On the other hand, the Restart parameter allows systems with small data buffers to decipher data which were enciphered in a single huge buffer -- provided the deciphering end can detect the start of the larger enciphering buffer.

We Can Use Message Keys (If Data Expansion is Acceptable)

For applications which could use more security, and which can accept data expansion, it is possible to use message keys. Basically, message keys are just random values which are enciphered before the data, and which are necessary to decipher the data proper. If the message keys are really random, there can be no statistical or known-plaintext attack on their enciphering.

A short but sweet form of message key can be implemented in DAGGER by explicitly setting the RNGencCur and RNGdecCur values in the DRT structure. The message key would be a one-word random value at some known location in the data frame (probably the start, to accommodate variable-length data frames). The sending end would encipher this value using some base key, and then would use the (unenciphered) message key word (which should be masked by AND with DRTRANDLAST to keep the value inside RandAR) to explicitly set RNGencCur before enciphering the rest of the frame. The deciphering end would decipher the message-key word with the base key, use it to set the deciphering RNGdecCur value (again masking first by AND with DRTRANDLAST), then decipher the rest of the frame. This would provide a 12-bit message key, which is not great, but would be fairly fast, and would seem to be a very serious obstacle to any "chosen plaintext" attack.

A more effective but certainly more imposing form of message key can be implemented by sending a larger amount of random data under the base key. Ideally, we might send some unique permutation of the 256 byte values. By enciphering this message-key data before the real data, we randomize the state of the Dynamic Substitution table before the real data are encountered. Note that this can be a substantial data expansion.

Yet another form of message key would simply append message key material to the base key, and then run SetUp on that resulting key. This will accommodate message keys of arbitrary size, but will require a fairly-lengthy initialization on each message key.

All forms of message key can be used simultaneously.

Terry Ritter, his *current address*, and his *top page*.

Last updated: 1996-05-31

The DAGGER Design

An Extremely-Fast Commercial Cipher Engine

Terry Ritter

DAGGER is a set of ciphering routines which add serious information security to a larger overall program. DAGGER is available in "portable C" and 80x86 assembly language.

DAGGER is a unique example of a *re-originating* *stream cipher*, a design which benefits greatly from our patented Dynamic Substitution technology. DAGGER is specifically intended to provide very high performance along with significant strength, and low impact on the rest of the system.

DAGGER handles variable-length data and does not expand data at all, and so can cipher variable-length database records "in place." Since DAGGER can "re-originate" on each ciphering, it can handle LAN data frames or network packets at the data-transport level where they may arrive out-of-sequence. Of course, DAGGER can also be used for conventional application-level ciphering, and the exact same cipher can be used for high-performance disk ciphering as well.

As a trade-off for extreme speed, DAGGER is constructed in a way known to be vulnerable to *defined*-plaintext attack. Thus, DAGGER is most appropriate for situations where an Opponent *cannot conduct* a defined-plaintext attack, and many such situations exist. Appropriate situations include local storage protection and most forms of end-to-end session ciphering (since intermediate nodes see at most *known*-plaintext).

DAGGER would be appropriate in a star topology network using a different key for each user. In a more general network, DAGGER can be appropriate given centralized delivery of session keys. DAGGER is less appropriate for network hops which carry messages from many users over a long time without changing keys.

Contents

- Goals
  - Extreme Speed, Commercial Strength, Broad Applicability, Superior to Realistic Alternatives
- Constraints
  - Variable-Length Data Frames, Absolutely No Data Expansion, Require No Information Beyond the Data Itself, Allow Data Frames to be Deciphered in Random Order, Protect Against "Known Plaintext" Attack
- A Trivial Alternative
  - Simple Confusion..., Is Easily Unconfused..., And Only Protects Against the Least Threat, Confusion is Not Protection
- Serious Alternatives
  - Block Ciphers, The Conventional Stream Cipher, The Conventional Combiner and Known-Plaintext, Confusion Sequence Re-Use
- The Dynamic Substitution Combiner
- General DAGGER Design
  - RNG Table, Multiple and Nonlinear Combiners, The Strength of Table-Confusion, DAGGER Encipher Architecture, DAGGER Decipher Architecture
- User Keys
  - Key Phrases, Random Data Keys
- Variable Strength
  - Expanding a Small Number of Bytes, Automatic In-The-Field Upgrade Potential, Potential Strength Selection Abuse, Finite Strength
- Key Expansion
  - Key Expansion by RNG, Jitterizer Nonlinearization
- Potential Attacks
  - RNG Table Reconstruction, Brute-Force, Ciphertext-Only, Known-Plaintext, Defined-Plaintext, Implications of a Defined-Plaintext Break, Defined-Key, Procedures and Practices Attacks
- Efficiency
  - Strength: The Ability To Resist Attack, Strength Versus Cost
- References

Goals

Any real cipher design must necessarily be a compromise. Ideally, a cipher could never be broken, would be infinitely fast, and would impose no requirements upon the rest of the system. This is clearly impossible. Therefore, we must be content with slightly more modest achievements.

1. **Extreme Speed**
   Paper cycle-counts of the DAGGER assembly-language inner loops shows about 27 cy/Byte for encipher, and about 40 cy/Byte for decipher on a 486. Consequently, in-memory ciphering could exceed 2.5M Bytes/sec on a 100 MHz machine. This is extraordinarily fast, and fast ciphering intrudes least on normal operations.
2. **Commercial Strength**
   In practice, it is nonsense and pointless to talk about "an unbreakable cipher." Indeed, a technical "break" of a serious internal or corporate cipher is probably the *least* likely way for information to be exposed; it is far more likely that a hard-copy would be found, stolen, or sold. In reality, a cipher need only be stronger than the easiest other way to acquire the same information. DAGGER is not intended to be the strongest possible cipher, but is instead an excellent compromise which provides both extreme speed and significant commercial-strength security.
3. **Broad Applicability**
   DAGGER can handle fixed-size or variable-size data, and data frames which are in-sequence or out-of-sequence. Thus, DAGGER can add security to most existing data storage or communications designs *without* placing new requirements on that system. This means that existing systems do not have to be completely re-designed for cryptography.
4. **Superior to Realistic Alternatives**
   It is easy to claim many things about a cipher which exists in a vacuum. The value of the DAGGER design should be measured by comparison to actual alternatives for the same applications.

Constraints

To make a general design which does not impose on existing systems, various strange requirements are necessary:

1. **Handle Variable-Length Data Frames**
   Theoretically, a DAGGER data frame can be virtually any size, and every frame can have a different size. This allows the cipher to function in a system which transports data in varying-size chunks, as well any other system which transports fixed-size frames.
2. **Produce Absolutely No Data Expansion**
   Under DAGGER, enciphered data will directly replace plaintext data -- and vice versa -- without ever requiring even one additional byte of storage.
3. **Require No Information Beyond the Data Itself (and a key, of course)**
   Repeatedly enciphering different data in exactly the same way is a cryptographic weakness in a stream cipher. Normally this weakness is handled by requiring some information specific to each particular data frame -- like a frame sequence number or message key -- so that the cipher can start out differently for each data frame. But these options either require that the system furnish some information beyond the data, or that the cipher expand the data, both of which are prohibited. By avoiding these requirements, DAGGER can avoid forcing the re-design of existing systems.
4. **Allow Data Frames to be Deciphered in Random Order**
   DAGGER supports random-access ciphering of database records and disk sectors. DAGGER also supports ciphering at the lowest level of data communications, where frames may need to be re-sent out of sequence to correct transmission errors.
5. **Protect Against Known Plaintext" Attack**
   We assume that the "opponent" can monitor and store many or most ciphertext transmissions, and will also be able to "obtain" (or guess) some amount of generally-innocuous but associated plaintext to help analyze the cipher.

Within these very significant Constraints, DAGGER was designed to provide a good tradeoff between strength and speed.

A Trivial Alternative

Instead of using a serious cipher, it is always possible to simply confuse the data to hide it from casual snooping.

Simple Confusion...

One way to hide data from casual snooping is to add some constant value to each character or byte value in a file. This can be very fast, and is probably sufficient to protect a file from nine out of ten people who might want to look at the it.

...Is Easily Unconfused...

Unfortunately, even if only one out of ten could penetrate the confusion, this could be a lot of people. In certain groups of ordinary law-abiding people (such as programmers), those who could reverse the confusion might even be a majority. Worse, the task of writing a program to reverse the confusion might be a hour's activity, and with such a program anyone could unconfuse files at will, in almost no time at all.

...And Only Protects Against the Least Threat

Worse, the nine out of ten who could not penetrate the confusion are almost certainly the most innocent and the least threat. The people we really have to worry about are those few who actively prey on information, and who can cause significant damage and liability. Certainly no data-theft criminal would have any problem at all penetrating simple file confusion.

Confusion is Not *Protection!*

Simple confusion is certainly *fast*, but not widely used. This is because simple confusion is effective only against those who pose the least threat, and is totally ineffective against any real attack. Thus, simple file confusion cannot be regarded as a prudent way to protect someone else's data.

Serious Alternatives

Within cryptography there are *codes* and *ciphers*. Codes deal with information at the level of words or phrases, and so are difficult to apply to arbitrary binary data. In contrast, ciphers deal with individual symbols (for example, bytes) or groups of symbols.

There are basically two different types of cipher: *block* ciphers and *stream* ciphers. Conventional block ciphers accumulate data in a fixed-size block and encipher the whole block as a single unit. In contrast, stream ciphers encipher data symbol-by-symbol (typically byte-by-byte).

The two types of cipher are distinct: To be effective, a block cipher generally must deal with blocks which are large enough to prevent analysis, typically 8 bytes or larger. Thus, a stream cipher which works on bytes really cannot be understood as a small block cipher.

Conventional Block Ciphers

Because block ciphers work on fixed-size blocks, they have problems handling continuously-variable amounts of data. As long as a full block can be filled for enciphering everything works well, but if there is some data left over -- less than a full block -- that data must be expanded to a whole block for enciphering. (Some designs do switch to stream-cipher mode for the last block only.) Variable-sized data is required by Constraint 1, and data expansion is disallowed by Constraint 2, so conventional block ciphers are not usable under these Constraints.

The Conventional Stream Cipher

The conventional stream cipher uses some form of addition to combine data with confusion values generated by some pseudo-random mechanism. Normally, a fairly complex pseudo-random generator can produce a sequence which is hard to predict, so the resulting cipher can be strong. But we cannot expect to operate a cipher in just any arbitrary way and yet still retain all the strength of which a cipher is capable. Most ciphers have weaknesses, and so are used in ways which hide or minimize those weaknesses. A conventional stream cipher consists of a confusion generator and a combiner: Both parts are potentially vulnerable.

The Conventional Combiner and Known-Plaintext

The usual stream cipher combiner is the simple exclusive-OR function (addition mod 2). Unfortunately, an exclusive-OR combiner is weak because it is so easily reversed: If someone can come up with some of the original plaintext, and then find the corresponding ciphertext, they can easily recover the original confusion sequence. Given some amount of the confusion sequence and a knowledge of the design of the confusion generator, an opponent may be able to fully-define the state of the confusion generator, and so "break" the cipher. This is the "known plaintext" or "probable plaintext" attack, and is very effective as well as being very well known.

Confusion Sequence Re-Use

Normally, a stream cipher is arranged to "never" re-use the confusion sequence. This can be accomplished by using a confusion generator with an astronomical sequence length, and by using a "message key" so that each and every enciphering will start at a different point in the sequence. Unfortunately, Constraints 2 (no data expansion) and 3 (no other information) prevent the use of message keys. Worse, Constraint 4 allows deciphering in random order, which apparently means that each and every ciphering must start at the same point in the confusion sequence. When a conventional stream cipher repeatedly uses the same part of its confusion sequence, knowledge of that sequence (from whatever source) will be sufficient to decipher subsequent messages. Thus, because of the given Constraints, the previously formidable stream cipher has become almost useless.

Re-using the same confusion sequence is a very serious problem for stream ciphers. In fact, this problem is so serious that, for most authors, it is almost unthinkable. Consider the comments made by Davies and Price in their 1984 text on computer network security:

> It is generally recommended that a running key be used only once for encipherment, because certain cryptanalytic attacks can take advantage of repeated use. [3:25]
>
> ...it is important to avoid using the same key for more than one message.... Use of the same key in encipherment for a number of plaintexts allows attacks either by a Kerckhoffs' superimposition or by elimination of the key by an exclusive-OR combination of the two ciphertexts. [3:40]

Similar comments come from Meyer and Matyas, in their 1982 text on cryptography:

> The stream cipher must not start from the initial conditions in a predictable way, and thereby regenerate the same cryptographic bit-stream at each iteration of the algorithm. In other words, the stream cipher must not reoriginate. [6:56]

And also Beker and Piper (in their 1982 text on stream ciphers), in the context of a confusion sequence produced by a linear feedback shift register (LFSR):

> If a modulo 2 adder is used to obtain the ciphertext from the plaintext and the shift register sequence, then any one of the three sequences is the modulo 2 sum of the other two. So if the interceptor knows the plaintext equivalent of m consecutive positions of the ciphertext then he knows m consecutive elements of the shift register sequence. ...if m = 2n then the interceptor can easily work out the entire key. [1:206]

But we are *forced* by Constraints 2 (no data expansion), 3 (no other information) and 4 (decipher in random order) to have our cipher at least *start out* in the same state. Thus, it will take a very special design to handle this situation.

The Dynamic Substitution Combiner

Because one of the main weaknesses of a conventional stream cipher is its simple and easily reversed exclusive-OR combiner, we might think to simply use a better combiner. The best, and perhaps the *only* serious alternative is Dynamic Substitution [4,5].

In the typical Dynamic Substitution combiner, each character is translated through a substitution table. Then, after each translation, the just-used entry is exchanged with some entry selected by a confusion value; this changes the "state" of the translation table.

Thus, whenever a particular plaintext character is translated, the translation for that character changes, and the more often a character appears, the more often the translation is changed.

Dynamic Substitution gives us a non-linear transformation which changes through time. The transformation is balanced, so that it is difficult for an attacker to separate plaintext data from confusion. And Dynamic Substitution is not directly subject to a simple known-plaintext attack, or at least not in the same trivial way as exclusive-OR.

But Dynamic Substitution is a tool, and not a panacea. Simply using an improved combiner in a situation which re-uses the confusion sequence is not likely to produce a very strong cipher. For example, someone with access to the system could encipher a lot of messages, and, simply by using each possible character, develop the original state of the substitution table. This could be done character-after-character, and thus establish every change in the table. The changes in the table would implicitly reveal a fixed confusion sequence. Then, given the original table and the known confusion sequence, the cipher is broken. Thus, even a Dynamic Substitution stream cipher is vulnerable to Constraint 5 (known-plaintext attack), if the same confusion sequence is used repeatedly.

General DAGGER Design

Because of Constraints 1 (variable-size data) and 2 (no data expansion), a conventional block cipher is not acceptable. Because of Constraints 3 (no other information) and 4 (decipher in random order) we apparently *must* re-use the confusion sequence, which should make a conventional stream cipher unacceptable. Fortunately, we can improve on the conventional stream cipher.

RNG Table

If we are going to re-use the same confusion sequence (for every message ciphered under a single key), there clearly is little reason to re-generate that sequence every time. We might as well save it in a table and just step through it. This can be very fast. It also opens up other alternatives.

Once we have the confusion sequence in a table, we can modify access to that table. Instead of just running through the same sequence every time, we can step through the table in random steps. For example (assuming a conventional stream cipher), we might think to add the ciphertext value to a current table pointer to select the next ciphertext value. But the ciphertext value is obviously exposed to any opponent, and this would reveal the distance between each confusion value used by the cipher. Since we must always start out in the same place, the opponent could know the input data, the corresponding ciphertext, and the distances between table values. One would expect such a system to fall fairly quickly.

Multiple and Nonlinear Combiners

To prevent the table offset value from being exposed, we might think to use more than one combiner. Of course, having multiple exclusive-OR combiners in sequence are not going to be much help, because they can be treated as a single exclusive-OR under a "known plaintext" attack. So we are pretty well forced into using at least one nonlinear combiner, and that pretty much means Dynamic Substitution.

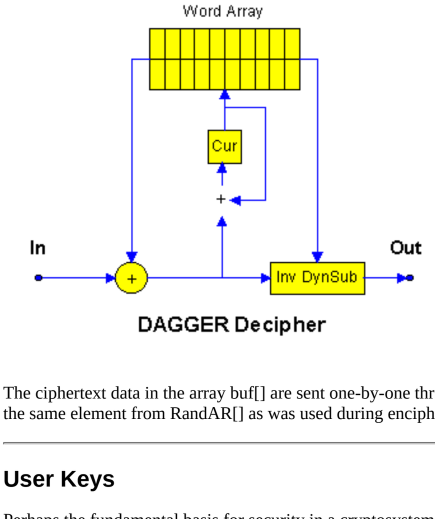
By using two combiners in sequence, first Dynamic Substitution and then exclusive-OR, we can take the value between two combiners as the RNG table offset value. Two bytes from the table will be required for every byte enciphered: One byte for the Dynamic Substitution combiner, and another for the exclusive-OR combiner. As long as the confusion value used by the exclusive-OR combiner is unknown, so is the RNG table offset value. And "known plaintext" cannot be used to attack the exclusive-OR, because the value out of the Dynamic Substitution combiner (and into the exclusive-OR) will not be known.

The Strength of Table-Confusion

In the past, the use of a table to generate the confusion sequence for a stream cipher would have been considered laughably weak. But the real weakness of such a design depends upon the ease with which the table can be reconstructed by a cryptanalyst. Previous stream ciphers could only use a single trivial linear combiner, thus allowing a "known plaintext" attack to completely reveal the original confusion sequence. But we can use a Dynamic Substitution combiner.

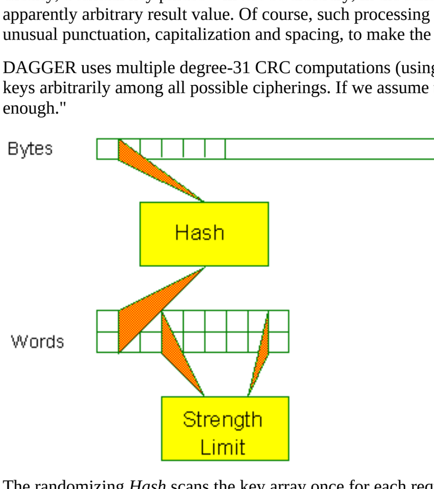
While it is not possible for Dynamic Substitution to have a "strength" independent of the rest of a cipher, it *is* possible to compare particular attacks against particular designs. When we use an RNG table, the "known plaintext" attack is trivial and absolutely effective against an exclusive-OR combiner. The exact same attack is not sufficient when the combiner is Dynamic Substitution.

DAGGER Encipher Architecture



The plaintext data in the array buf[] are sent one-by-one through Dynamic Substitution and exclusive-OR and the resulting ciphertext is placed back into buf[]. Each combiner uses a byte from the RNG table as selected by index value RNGCUR. The value produced by the Dynamic Substitution combiner is added to RNGCUR after each byte is ciphered. Note that Dynamic Substitution operates directly upon plaintext data; this allows the dynamic combiner to respond to statistical variations in the plaintext.

DAGGER Decipher Architecture



The ciphertext data in the array buf[] are sent one-by-one through exclusive-OR and then Inverse Dynamic Substitution, and the resulting plaintext is placed back into buf[]. Each combiner uses exactly the same element from RandAR[] as was used during enciphering; this happens because both modes start at the same place in the RNG table, and update RNGCUR with the same intermediate value.

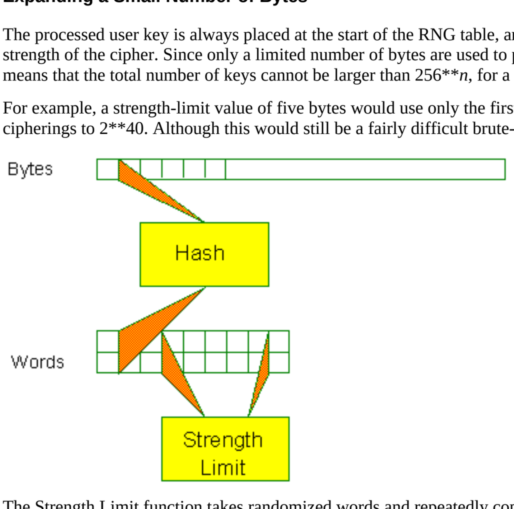
User Keys

Perhaps the fundamental basis for security in a cryptosystem is the expectation that each different user key will produce a different -- arbitrary -- enciphering. The original plaintext hides like a needle in a haystack of other keys. But if the key is for some reason obvious or apparent, we can expect an opponent to try that first, and so penetrate the cipher with less than the expected effort. At a minimum, everything depends upon having extremely unique or arbitrary keys.

Key Phrases

Clearly, a textual key phrase will *not* be arbitrary, in the sense that some words follow others preferentially and some letters do likewise. Thus, it is important to process or "randomize" the key to some apparently arbitrary result value. Of course, such processing cannot compensate for a fundamentally obvious key, so we want to see long, strange phrases. Text keys should contain some numbers, and unusual punctuation, capitalization and spacing, to make the phrase different from anything ever placed on paper.

DAGGER uses multiple degree-31 CRC computations (using different polynomials) for randomizing the key. While this does not increase the "uncertainty" in a key, it does distribute the most likely keys arbitrarily among all possible cipherings. If we assume that normal text may have an uncertainty of perhaps 2 bits per byte, and that 80 bits is enough, key phrases of 40 characters should be "long enough."



The randomizing *Hash* scans the key array once for each required hash word, using a different CRC polynomial each time. The CRC results are saved in a word array for subsequent processing.

Random Data Keys

DAGGER uses exactly the same CRC mechanism to process any type of key, so a key server could deliver different sorts of values at different times. The ideal key would be some sort of "really random" or "physically random" binary value, but good, strong keys can be developed from an accumulation of second-by-second details (such as LAN usage), using CRC or a cryptographic hash. On the other hand, simple reliance on time-of-day -- even if cryptographically hashed -- would have to be considered insecure.

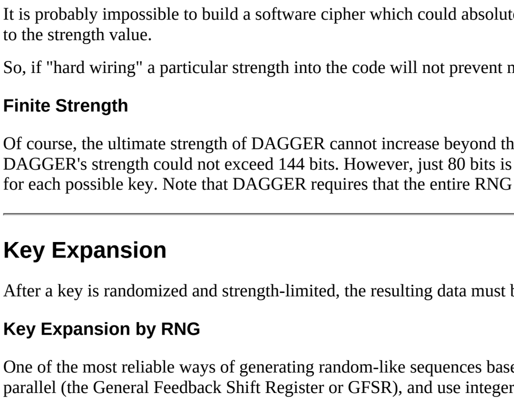
Variable Strength

Some government agencies seem interested in having ciphers which are used for international communication be weak enough to be broken by an agency with world-class computation resources. To help these people, DAGGER can be weakened to almost no strength at all, simply by selecting a low strength-limit value during start-up.

Expanding a Small Number of Bytes

The processed user key is always placed at the start of the RNG table, and will be expanded to fill the table. But before expansion, we can limit the number of unique bytes, and in this way vary the strength of the cipher. Since only a limited number of bytes are used to produce the expanded data, there can be only that much uniqueness, no matter how many bytes were previously in place. This means that the total number of keys cannot be larger than 256***n*, for a strength-limit value of *n* bytes.

For example, a strength-limit value of five bytes would use only the first 40 bits of randomized key material, no matter how much key was actually available; this would reduce the number of different cipherings to 2**40. Although this would still be a fairly difficult brute-force problem for an individual, it might be an acceptable value for a government agency with huge computing resources.



The Strength Limit function takes randomized words and repeatedly copies the first selected number of bytes to expand the data to a total of nine words (144 bits). Thus, 144 bits is the DAGGER "design strength."

Automatic In-The-Field Upgrade Potential

An interesting consequence of strength limitation is that the key-distribution mechanism could also distribute the strength-limit value. Thus, as time passes and government approval allows, the DAGGER cipher can be strengthened, in the field, *without software change*. Simply by sending a larger strength-limit value from the key server, the cipher could automatically be made stronger. No other changes would be required.

Potential Strength Selection Abuse

Would the strength parameter be liable to abuse? Note that many modern cipher systems are implemented in software, and any particular system remains as it is simply because programmers refrain from changing it. An equivalent strength value could, of course, be "hardwired" into the code. But the "users" of this package are not users in the normal sense: The users of this package are *programmers*. And programmers have access even to supposedly "hardwired" elements. Any programmer with access who wants to change the strength of a software cipher can do it. They can even do this *without* altering the cipher code proper simply by double-enciphering.

It is probably impossible to build a software cipher which could absolutely prevent double-enciphering; in practice, a simple agreement may suffice. And a similar agreement could be made with respect to the strength value.

So, if "hard wiring" a particular strength into the code will not prevent misuse, why ruin the advantage of flexibility for the vast majority of users who will not misuse the code?

Finite Strength

Of course, the ultimate strength of DAGGER cannot increase beyond the effort required for some effective attack which is not based on "brute force." Since the largest DAGGER key is nine words, DAGGER's strength could not exceed 144 bits. However, just 80 bits is generally thought secure against "brute force," although this obviously depends on the effort involved to generate the system state for each possible key. Note that DAGGER requires that the entire RNG table be developed -- *and jitterized* -- before the cipher can function.

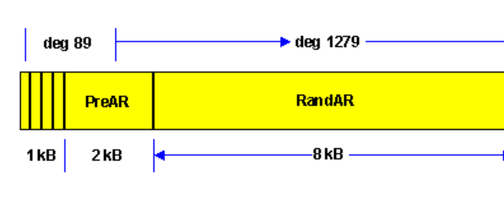
Key Expansion

After a key is randomized and strength-limited, the resulting data must be expanded to fill the whole RNG table. This is done using Additive RNG designs of increasing polynomial degree.

Key Expansion by RNG

One of the most reliable ways of generating random-like sequences based on an initial value or "seed" is the Linear Feedback Shift Register (LFSR). More recent mechanisms work on bytes or words in parallel (the General Feedback Shift Register or GFSR), and use integer addition instead of the mod-2 add or exclusive-OR (the Additive RNG) [8].

As described earlier, the DAGGER strength-limitation stage expands the randomized key to 9 words (18 bytes). From this point, DAGGER uses a sequence of Additive RNG [4:27] designs with different mod 2 primitive feedback polynomials [3:62-63; 1:359] of increasing size. Each takes two or more of the defined values, adds them, places the result at the next undefined position, then moves ahead one word. The last, a degree-1279 trinomial, uses two values from the preceding 1279 words to produce each result value. This happens repeatedly until the RNG array is filled.

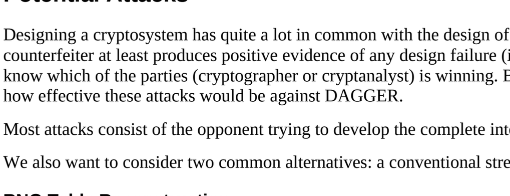


The *Expand* block takes the nine-word array produced by *Strength Limit* and expands it to fill the array. Each expansion polynomial uses two or more already-defined word values to produce a resulting word value until the array has been filled. The Expand output always leads in advance of the input locations.

Jitterizer Nonlinearization

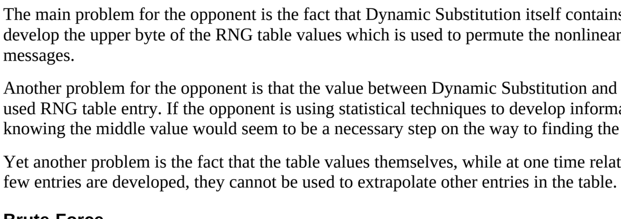
Each of the Additive RNG's is a linear mechanism; we would expect that if a cryptanalyst could somehow find as many elements as the polynomial degree, they could solve and reproduce the sequence. Of course, the degree-1279 polynomial implies that at least 1279 two-byte words would be required to initiate this part of an attack, and both halves of each word are used to encipher each single character, making identification difficult. But to further avoid such an attack, DAGGER uses a relatively-fast "jitterizer" mechanism [8:111], which steps through the table and deletes various numbers of bytes (the rest of the table is copied down) at various times. The jitterizer also randomizes each "take" group of values by exclusive-ORing one of the deleted values to each element. The whole process is fairly quick, and should prevent easy attacks on the table as a linear sequence.

On average, about 10% of the data are deleted from the table. This means that we must create more data originally, just in order to guarantee enough jitterized material for RNG table operation.



The *Jitterize* function takes the filled word array produced by Expand and deletes segments of the sequence at random by simply not-copying sections of the sequence. Note that the Jitterize output will always lag at or behind the current input location.

We can relate the resulting array to the fields used in the software mechanism:



PreAR, *RandAR*, and *ShufAR* are the various areas of the DRT structure defined to hold DAGGER state. This structure is about 13KB in size, with the *RandAR* itself consuming 8KB. The deg 9 polynomial fills only a tiny portion of the array to the left of "deg 89"; deg 89 fills into *PreAR*, and deg 1279 fills through the end of the structure. All of this happens before *Jitterize*, of course. Jitterizing deletes some of the table data and moves the rest down; something like the last 1KB of the DRT structure (about half of the *ShufAR*) will remain unchanged. The jitterized part of the *ShufAR* data is used to shuffle the Dynamic Substitution array located in one of the boxes at the start of the DRT structure.

The four small boxes on the left of the array represent 256-byte substitution boxes. Two of these are the forward original and inverse original which are used to reset the system for a new data frame. The other two boxes are the working forward and working inverse tables which change during processing.

Potential Attacks

Designing a cryptosystem has quite a lot in common with the design of a currency: As a currency designer tries to prevent counterfeiting, a cryptographer tries to prevent cryptanalysis. But a counterfeiter at least produces positive evidence of any design failure (if you can find it); in contrast, a cryptanalyst generally leaves no technical tracks. Because of the lack of feedback, it is hard to know which of the parties (cryptographer or cryptanalyst) is winning. But various sorts of well-known attacks have proven effective against other cryptosystems, so it only makes sense to think about how effective these attacks would be against DAGGER.

Most attacks consist of the opponent trying to develop the complete internal state of a cipher system in order to decipher other unknown messages enciphered under the same key.

We also want to consider two common alternatives: a conventional stream cipher, and also a cipher like DAGGER but without the exclusive-OR second combiner.

RNG Table Reconstruction

Perhaps the most obvious attack would be to aim at the use of the relatively small static table as the source of the confusion sequence.

The main problem for the opponent is the fact that Dynamic Substitution itself contains substantial state. Unless the opponent can develop or interpret that state, it would seem to be almost impossible to develop the upper byte of the RNG table values which is used to permute the nonlinear combiner. And only if *both* bytes of each RNG table entry are found can the opponent hope to break arbitrary messages.

Another problem for the opponent is that the value between Dynamic Substitution and exclusive-OR is unknown. This is a problem for two reasons: First, this value specifies the distance to the next used RNG table entry. If the opponent is using statistical techniques to develop information about a particular table entry, it would seem to be necessary to identify when that entry comes up again. Next, knowing the middle value would seem to be a necessary step on the way to finding the lower byte in a RNG table entry.

Yet another problem is the fact that the table values themselves, while at one time related in a linear way, have been processed by the jitterizer to no longer be linearly related. This means that, even if a few entries are developed, they cannot be used to extrapolate other entries in the table.

Brute-Force

The main form of brute-force attack consists of trying all possible keys in the hope of eventually finding one which deciphers the message. The designer should make the key-space large enough so this could be a long, hard search. Normally, an 80-bit key is large enough. Thus, ideally, we would like to see at least 80 bits of "uniqueness" in each of our keys. Since language is amazingly redundant, it is necessary to have much longer text keys than those which are random binary values. For best strength, we would normally like to see the strength-limit value as large as possible, and at least 10 bytes. If the strength-limit value is less than 10 bytes, opponents with substantial resources may be able to make brute-force an effective attack. **This is not a flaw**; it is the reason for having a strength-limit value.

Ciphertext-Only

A "ciphertext only" attack consists of having only the ciphertext and the design, and no other information. Raw ciphertext is one of the hardest ways to break a cipher, and yet is probably the way most of us mistakenly imagine that ciphers are broken. (Brute-force is also a ciphertext-only attack, but is considered separately because there is nothing particularly clever about it.)

Ciphertext-Only *vs.* DAGGER

It is not clear that there is any effective attack on DAGGER using ciphertext only.

Ciphertext-Only *vs.* A Conventional Stream Cipher

On the other hand, consider the conventional stream cipher: in our applications such a cipher necessarily must re-use its confusion sequence. Because it has only a simple exclusive-OR combiner, any two ciphertext messages (enciphered under the same key) can be exclusive-ORed to "cancel out" the confusion sequence. The result is a far simpler cipher: two plaintext messages combined by exclusive-OR. By doing this for a number of different messages (and combinations of those messages), character coincidences, especially space characters, can be located. Then other characters in those positions can be revealed, and we can fill in the blanks, just like a newspaper recreational cipher.

Known-Plaintext

The "known-plaintext" attack consists of "obtaining" or guessing some amount of plaintext and the corresponding ciphertext. Getting some of the plaintext may not be too hard to do, since most users occasionally cipher otherwise innocuous messages, and then see no need to dispose of those messages in a secure manner. Alternately, it may be possible to get the sender to encipher a particular message for some reason, perhaps a particular contract, for example. Graphic logos and signatures contain a huge amount of data which can be fairly easily obtained. And even if the full text of a message is not known, the messages may follow a particular format or pattern which can be anticipated, and some of the message words or phrasing may be guessed.

The known-plaintext attack is the major attack on conventional stream ciphers because it can be devastatingly effective. Given both the plaintext and the ciphertext for a given message, the conventional exclusive-OR combiner immediately reveals the confusion sequence. If that sequence were produced by a Linear Feedback Shift Register (LFSR), a relatively small amount of the sequence would suffice to define the LFSR and break the rest of the sequence.

Known-Plaintext *vs.* DAGGER

Fortunately, DAGGER is not limited to an exclusive-OR combiner, nor does it directly use an LFSR confusion sequence generator. In DAGGER, the Dynamic Substitution combiner produces some unknown middle value which the exclusive-OR processes into ciphertext. To find the confusion sequence one must assume some value from the nonlinear combiner, but any value is equally possible. Similarly, we could gain information about the nonlinear combiner if we could assume the confusion sequence value into the exclusive-OR, but, again, any value is possible.

Note that DAGGER uses two confusion bytes for every data byte enciphered. Certainly a single byte of ciphertext cannot possibly identify both confusion bytes, no matter what technique is used; there is simply not enough information contained in one byte to define two bytes. Therefore, any effective known-plaintext technique would likely have to be statistical or iterative in nature, a situation substantially different from the classical approach.

Known-Plaintext *vs.* Conventional Stream Cipher

Now consider the conventional stream cipher: In our applications such a cipher necessarily must re-use its confusion sequence, and has only a simple exclusive-OR combiner. With one suitably-long known-plaintext message, the confusion sequence can be immediately recovered simply by exclusive-ORing the plaintext and ciphertext. Then, any other message (under the same key) can be deciphered simply by exclusive-ORing that ciphertext with the now-known confusion sequence. This is an immediate and trivial operation, which is a complete "break" for this type of cipher, when used in our applications.

Known-Plaintext *vs.* DAGGER Without Exclusive-OR

Now consider DAGGER without exclusive-OR. Clearly, if the initial Dynamic Substitution table and the RNG table were known, the cipher would be broken. By examining many different messages, most or all the initial Dynamic Substitution table state (or the state after some fixed header) can be absolutely known. This leaves the RNG table.

Now we work on defining the initial part of the RNG table: Different messages will use the Dynamic Substitution table entries in different sequences, but the first time each entry is used, it should generally produce the previously-defined result. If it does not, this implies that that entry has been exchanged, and this both reveals a hidden confusion value and restricts to a few locations where that value must have occurred.

Since the table-step value is not hidden in this cipher, but is just the ciphertext value itself, we know immediately when a particular table element is re-used, and this must be a big help in reconstructing the table. In an average of only 32 characters, the RNG index will wrap around the 4k table, and then we have an opportunity to work on the initial table area again. Possibly, with enough messages, we may be able to mostly define that area. Then, given known RNG table values, we could extend the known Dynamic Substitution state beyond the first character, which would be the next step in defining the next area of the RNG table.

It is not currently known how much effort would be involved in a complete known-plaintext break of DAGGER without exclusive-OR, but the above scenario does not look good.

Defined-Plaintext

A "defined-plaintext" attack means that the opponent gets to specify messages which are enciphered under the unknown key and returned to the opponent. We assume that it is possible for the opponent to cipher huge numbers of messages in an attempt to get some information about the internal cipher state.

Defined-Plaintext *vs.* DAGGER

Consider a "defined plaintext" attack on DAGGER: Somehow, the opponent gets to encipher arbitrary messages under the target key. We assume that the key does not change during the attack.

1. 256 one-byte messages completely explore the initial Dynamic Substitution state, although the results are not clear because they are further confused by the XOR combiner and random value.
2. 256 more two-byte messages completely explore the next Dynamic Substitution state (after a given character is enciphered). Again, the results are not absolutely clear.

Note that the initial Dynamic Substitution state and next Dynamic Substitution state are related in that there are probably (and at most) two different entries, the result of a single exchange. So there are at least 254 entries in each state which are exactly the same, and related by exclusive-OR with the same single value. This value is the exclusive-OR "difference" between the initial and next exclusive-OR value, and should be easily developed.

Since we assume a defined plaintext attack, the input values are known absolutely; the RNG table values are supposedly hidden. But because we can find the two different (exchanged) elements in the Dynamic Substitution table, and know which element was selected by the input value in the previous step, we now absolutely know the "hidden" confusion value from the RNG table. Of course, we still do not know the RNG table step value, which is hidden by the exclusive-OR.

If we work interactively, it is not always necessary to scan the entire Dynamic Substitution table each time to find the single unknown exchange element. Sometimes this will be necessary, but other times we will find it quickly, so we can assume an average somewhat more than 128 tries. But if the attack is not interactive, all 256 possibilities must be checked.

In addition to the Dynamic Substitution table, we also know the exclusive-OR difference between the two states. Thus, we know the high byte of one RNG table entry, and also the low byte, in the sense that all our known low bytes will be exclusive-ORed with some particular other value. Since all low-bytes will be exclusive-ORed with the same "other value," there are only 256 possibilities. This also gives us the distance between entries, in the same exclusive-OR sense.

3. Suppose we repeat the above process many times. This will cycle around the RNG table and eventually land on RNG elements which have been used before. Each time we land on the same RNG element we will get the same result for both high and low bytes (after correcting for the new exclusive-OR difference). Since there are 64k possible two-byte values and only 4k elements, a particular two-byte value is likely (15 times out of 16) to identify a particular element. Thus, we can begin to build up an absolute knowledge of the high bytes, and we will have only 256 possibilities for the entire array of low bytes.

Suppose we are lucky and only have to do the identification exactly 4k times to find each entry in the RNG table; this means that we incur the cost of at least 4k x 128 = 512k messages. (Then we will finally have to try an average of 128 possible exclusive-OR offsets to find the correct low-byte values.)

But the table is still unordered. The absolute Dynamic Substitution output is not known, and this is the RNG element increment. We do know that each next element is somewhere in the array from the last element to 255 elements further on. Thus, we can try to group each element, in the sense that from a particular known element, another known element is at most 255 elements away. Probably if we go through the table many times (as a guess, 128 times), we may be able to generally group and order the entire table. This means 65M messages.

4. But what we really want is the Dynamic Substitution output value. With that and the element values, we can order the table and completely recover the entire cipher state. Now note that there are only 256 possible unknown exclusive-OR offsets; one of them gives us each correct intermediate value. Thus, after we identify a few repeated elements (and have the sequence of intermediate RNG step values, each offset by the same exclusive-OR value), we can identify the correct exclusive-OR offset, and then order the table by RNG step. Finding 4k RNG table elements would require 4k scans, or 4k * 128 = 512k messages. So, if the cipher system allows a remote node to define 512k messages for enciphering under a single key, it risks being broken.

Under an interactive defined-plaintext attack, something like 512k specially-defined messages could produce a complete break. Of course, something less than that number could produce a partial RNG table, so that bits and pieces of plaintext could be recovered (which is probably not particularly useful unless the RNG table is almost completely defined).

Defined-Plaintext *vs.* Conventional Stream Cipher

There is no need to further consider the conventional stream cipher, which was completely broken under the far easier known-plaintext attack.

Defined-Plaintext *vs.* DAGGER Without Exclusive-OR

Now consider a similar attack on DAGGER without exclusive-OR. In this case, the Dynamic Substitution output is the ciphertext output and also the RNG table increment. This would support the easy and straightforward reconstruction of the RNG table, as well as identification of element re-use. This would avoid the need for any technique to order the RNG table, but would still apparently take about 4k scans (512K messages).

A better alternative would seem to exist in defining messages which are particularly useful in extending the logical sort of attack described under known-plaintext. This should take substantially less material.

Implications of a Defined-Plaintext Break

Note that a "defined plaintext" attack requires the target key to encipher a huge number of specially-constructed messages from the opponent (and the ability to identify the resulting ciphertext). Such an attack generally could not be applied to data storage ciphering, because the target key would not be present during the attack.

For LAN ciphering, it is important to recognize a possible security risk if the system allows remote nodes to submit huge numbers of arbitrary messages for enciphering under a single local key. One possibility would be to absolutely prevent any such ciphering, another would be to limit the number of such messages enciphered under a single key. The universal use of short-term session keys (no week-long sessions!) would also help to minimize this risk.

If the opponent does manage a "defined plaintext break," then she can read any messages enciphered under the broken key. Messages enciphered under another key are still hidden; the attack must be repeated for each new target key. To reduce the amount of data enciphered under a single key, we should use more keys, each for a shorter period. A secure LAN would generally use a "key server" to provide new random keys for each session.

The potential threat of a defined-plaintext attack can be essentially eliminated through the use of message keys, which are completely-random values transmitted with each message. However, message keys would violate Constraint 2 (no data expansion) or 3 (no other information) in our proposed applications. In a sense then, it is these particular requirements which open windows of vulnerability in the cipher. It is instructive to compare the vulnerability of the DAGGER design to other alternatives under the exact same Constraints.

Note that this cipher weakness applies exclusively to a "defined plaintext" attack. The approach does not apply to arbitrary messages, even if millions of messages of "known plaintext" (associated plaintext and ciphertext) are available. "Defined plaintext" requires a particular type of strongly-related messages as an "attack suite."

Defined-Key

In a "defined-key" attack, the opponent chooses keys for their particular effects. If these effects are in any way "linear" and the key bits independent, it may be possible to develop the correct key bit-by-bit, and so break the cipher.

Because in DAGGER the keys are processed by multiple CRC polynomials, virtually every resulting bit depends upon each and every bit of the external key. Thus, it seems rather unlikely that any defined-key attack could succeed.

Because the other ciphers under discussion (conventional synchronous stream cipher, and DAGGER without exclusive-OR) do not define the key-expansion stage, we have nothing to say about a defined-key attack on them.

Procedures and Practices Attacks

Even a cipher which is technically secure can be used in ways which make it easy to penetrate security. Such use could even confuse the situation, laying the fault at the door of the cipher itself, when the real fault lie in procedures and practices.

Obtaining a paper copy of the plaintext, either before sending or after reception, is an ideal way to "break" a cipher. Using Electro-Magnetic Field (EMF) emissions to reproduce a CRT screen showing the plaintext is also an excellent way to "break" a cipher. Gaining access to a computer with the plaintext inside, or planting a "virus" program to obtain that plaintext are also practical "attacks." Or perhaps we could just bribe someone who had access.

The extent of such possible attacks is so great that it is almost impossible to rule them all out, and this is yet one more reason why a realistic cipher does not need to be "absolutely unbreakable." Not only could we never prove such a thing, nor afford the processing needed to use it, but the usual procedures and practices of business provide any number of insecurities that cryptography cannot touch.

In practice, a cipher only needs to require substantially more effort to "break" than would be necessary to compromise security in some other way.

Efficiency

In software, all functionality requires processing, and this requires time. The fastest possible cipher is no cipher at all, but such a cipher is not very effective at hiding information. Conversely, ciphers which are designed to be extremely strong inherently demand substantial amounts of processing. Our goal here is to get "the most bang for the buck."

Strength: The Ability To Resist Attack

The "strength" of a cipher is the ability to resist attack. Unfortunately, there is no external test which can tell how strong a particular cipher design really is. What we can do is to look at the design with respect to the various known attacks, and try to see how difficult and effective they would be. It is certainly much more difficult to mount a successful "known plaintext" attack against DAGGER than it would be against a conventional exclusive-OR stream cipher designed for the same application.

Currently, the only known effective attack on the DAGGER design is a lengthy defined-plaintext attack. That said, DAGGER is a relatively simple cipher (as these things go), and may be vulnerable to the extent that an opponent can reconstruct the RNG table.

Strength Versus Cost

The issue in the DAGGER design is *not* absolute strength: The issue is: **maximum strength for minimum execution cost**. A direct comparison between DAGGER and a conventional stream cipher -- *operating under similar Constraints* -- will show a significant strength advantage for DAGGER, at a relatively modest cost. DAGGER is exceptionally fast and comparatively strong.

References

1. Beker, H. and F. Piper. 1982. *Cipher Systems*. Wiley.
2. Bright, H. and R. Enison. 1979. Quasi-Random Number Sequences from a Long-Period TLP Generator with Remarks on Application to Cryptography. *ACM Computing Surveys*. 11(4): 357-370.
3. Davies, D. and W. Price. 1984. *Security for Computer Networks*. Wiley.
4. Golomb, S. 1982. *Shift Register Sequences*. Aegean Park Press.
5. Knuth, D. 1981. *The Art of Computer Programming, Vol. II: Seminumerical Algorithms*. Addison-Wesley.
6. Meyer, C. and S. Matyas. 1982. *Cryptography: A New Dimension in Data Security*. Wiley.
7. Ritter, T. 1990. Substitution Cipher with Pseudo-Random Shuffling: The Dynamic Substitution Combiner. *Cryptologia*. 14(4): 289-303.
8. Ritter, T. 1990. Dynamic Substitution Combiner and Extractor. *U.S. Patent 4,979,832*.
9. Ritter, T. 1991. The Efficient Generation of Cryptographic Confusion Sequences. *Cryptologia*. 15(2): 81-139.

Terry Ritter, his current address, and his top page.

Last updated: 1996-09-17

Penknife Features

A data security program with key management for e-mail, for use under Microsoft Windows or DOS.

Contents

- Overview
- Operation
- Examples of Commands
- Features
- Key Management

Overview

Penknife takes any secret key phrase, and transforms or *enciphers* files into lines of jumbled ASCII text. The original file contents can be recovered from the jumble only by *deciphering* with exactly the same key. Files thus protected can be saved locally, archived off-site, or sent by e-mail without exposing their contents.

Penknife can automatically skip e-mail headers and signatures when deciphering, minimizing the need for manual "fix up." Enciphered alias files hold secret keys and allow them to be selected by public aliases, thus minimizing the impact of secrecy on ordinary users. Keys can be updated while users can continue to use exactly the same alias in exactly the same way. Dated alias allow access to old archived ciphertext protected by outdated keys.

Now available in Advanced, Commercial Demo, Decipher-Only and Corporate versions for DOS and Microsoft Windows. Not for export.

Operation

For greatest ease and security, the user should have an *alias file*. An alias file is an enciphered file of public aliases and related secret keys. The user can thus select one of many secret keys by supplying the key to the alias file plus the public alias for the desired key.

In normal operation, the user supplies the input filename, output filename (if different), the alias, and selects encipher or decipher. The program requests entry of the alias-file key, once. The program then finds the "closest" alias file, deciphers it in memory only, finds the indicated alias, then uses the secret key associated with that alias to cipher data.

The "generate" function in the advanced version automatically creates or adds to a local alias file, and will encipher the new key in a different file for transport. The transport file can be used as a beginning alias file, or added to an existing alias file, without deciphering the alias file.

Examples of Commands

- **Encipher Multiple .TXT Files Using Alias "fred"**

```
penknife  *.txt  *.pen  /e  /a fred
```

- **Decipher Multiple .PEN Files In-Place Using Alias "fred"**

```
penknife  *.pen  /d  /a fred
```

- **Decipher Using Key Active on Dec. 15, 1993**

```
penknife  file1.pen  file1.res  /d  /a fred  /m 93-12-15
```

- **Change Key to Alias File**

```
penknife  penknife.mgt  /d    (Enter Old Key)
penknife  penknife.mgt  /e    (Enter New Key)
```

- **Encipher Particular Files into Ciphertext Archive**

```
penknife  file1.txt+file2.txt+file3.txt  arch1.pen  /e
```

Features

- **Easy to use** under Microsoft Windows or DOS.
- **Fast:** About 80K bytes/sec (on a 486DX2/50).
- **Strong:** Uses a 63-bit internal key, with a random 32-bit line key on each ciphertext line.
- **Small:** Under 50K including on-line help.
- **Enciphers any file of any sort and recovers the original data without loss.** CRC error-detection checks each deciphered file. CRC also detects the use of a wrong deciphering key.
- **No "wrong" operating mode:** produces only network-transportable ASCII ciphertext.
- **Transparently handles DOS or Unix text lines.**
- **Will ignore e-mail "headers" or ".sigs"** or optionally pass them through to keep with the deciphered text.
- **Can limit output files to under 48K** for Internet transmission.
- **Optionally overwrites the original file,** thus (providing DOS cooperates) hiding the original data, even from file-recovery programs.
- **Ciphers one file, multiple files, or an entire disk with a single command.** Filenames being selected can be simply displayed without ciphering.
- **Supports enciphered batch files** of commands.
- **Ciphertext can be concatenated** in secure archives.
- **A limited commercial demo** can be distributed for corporate evaluation or individual use.
- **The straightforward secret key cipher** is much like using house keys or car keys.
- **Uses patented Dynamic Substitution technology.** Does not infringe any known patents.

Key Management

- **Enciphered alias files** for each user hold and protect their secret keys.
- **A public alias for each key** selects that key from among all others in the alias file.
- **A user need only remember one key** for their alias file, instead of remembering every key they use.
- **Aliases** support the use of large random keys for better security.
- **Dated aliases** support access to old ciphertext.
- **Aliases can be kept on a floppy** and personally retained when not in use.
- **A key-generation mode** constructs a long random key for a given alias. Groups can be given a single key for all members to use with each other.
- **Generated keys are automatically added to the local alias file.**
- **Generated keys are also placed in a separate enciphered file** for transport to the far end or for distribution to other members of the group.
- **New keys can be added to an alias file without deciphering the file** and thus exposing the keys inside.
- **Key updates restore security** periodically, or when individuals leave a group.
- **Painless key-update:** The user continues to use exactly the same alias to select a new key.
- **Key-updates can be programmed in advance,** to support automatic company-wide shifts to new keys on a given date.
- **Corporate version supports corporate key control.**

Terry Ritter, his current address, and his top page.

Last updated: 1995-08-07

The Penknife Cipher User's Manual

File Encryption and Key Management for Electronic Mail

Terry Ritter, P.E.

Ritter Software Engineering
2609 Choctaw Trail
Austin, TX 78745
(512) 892-0494

Copyright 1993 - 1995 Ritter Software Engineering
Protected by U.S. Patent 4,979,832

Introduction

Penknife is a serious stream cipher for e-mail based on Dynamic Substitution technology. Penknife will encipher any file of any type into a file of text lines, and will decipher that ciphertext into a file which is byte-by-byte the same as the original. Penknife is currently implemented as a DOS program with an internal command line which works well under Microsoft Windows.

Penknife users normally select a hidden random key from an alias file of such keys (the alias file might be provided by the local corporation). A particular key is selected by an associated alias tag which need not be secret. This allows the hidden keys to change (e.g., as employees leave), while the user continues to use the exact same alias.

Penknife is a "conventional" or "Secret Key" cipher, and is ideal for secure local storage or secure communication within a corporation. Although "Public Key" ciphers are often suggested as an alternative, for true security, they require that received keys be *certified*. When this poorly-understood step can be ignored by users, supposedly secret information can be exposed to outsiders. This sort of problem does not occur in a Secret Key cipher.

Contents

- Start Up
  - Overview
  - The Penknife Versions
    - Advanced
    - Commercial Demo
    - Decipher-Only
    - Corporate
  - Installation
    - DOS Installation
    - Microsoft Windows Installation
    - What To Expect
- Simple Operation and Help
  - Help
  - Simple Enciphering
  - Simple Deciphering
  - Sending Large Files by E-mail
  - Recovering Large Files from E-Mail
  - Creating a Key-Alias
  - Installing a Tranported Key-Alias
  - Using a Key-Alias
- OPTIONS, Commands and Features
  - Command-Line Options
    - /Alias <tag>
    - /Batch
    - /Decipher
    - /Encipher
    - /File-alias
    - /Generate
    - /Key <keycharacters>
    - /Limit
    - /Msgdate <date>
    - /Nocipher
    - /Passthrough
    - /Quiet
    - /Tree
    - /Zmode
    - /7bit
  - Commands for Common Operations
    - Simple Enciphering
    - Simple Deciphering
    - Sending Large Files by E-mail
    - Recovering Large Files from E-Mail
    - Creating a Key-Alias
    - Installing a Transported Key-Alias
    - Using a Key-Alias
    - Encipher File "in place"
    - Decipher File "in place"
    - Generate Alias Key Effective Jan. 1, 2001
    - Wildcard Alias-Encipher .TXT Files into .CLO Files
    - Decipher File Using Alias "fred"
    - Decipher In-Place and Pass-Through Plaintext
    - Encipher Using Alias Key Active on Dec. 15, 1993
    - Decipher Using Alias Key Active on Dec. 15, 1993
    - Encipher a File for File-Alias Deciphering
    - Wildcard File-Alias Deciphering
    - Wildcard File-Alias Deciphering with Alias "fred"
    - Change The Alias-File Key
    - Encipher Multiple Files into Ciphertext Archive
    - Encipher Particular Files into Ciphertext Archive
    - Decipher Ciphertext Archive
  - Penknife Features
- Detailed Operation
  - Two-Filename Mode
  - One-Filename Mode
  - The User Key
  - Alias Files
  - Penknife Batch Mode
  - Other Penknife Features
    - Append Mode
    - Internet Size Limits
    - Wildcard File Specification
    - Directory-Tree Scanning
    - Append Operator (+)
    - Sequence Operator (,)
    - Z-Mode
  - Cipher Technical Overview
  - One-Filename Detailed Operation
  - Sample Commands
    - One-Filename Simple Ciphering
    - Two-Filename Simple Ciphering
    - One-Filename Alias Ciphering
    - Two-Filename Alias Ciphering
    - One-Filename Alias Wildcard Ciphering
    - One-Filename Alias Wildcard Ciphering and Directory-Tree Subdirectory Scanning
    - No Ciphering, Just a File-Match Display
    - Two-Filename Alias Wildcard Ciphering and Directory-Tree Scanning
    - Two-Filename Alias Wildcard Ciphering with Specific *ToFile:* "Append Mode"
    - One-Filename File-Alias Ciphering
    - One-Filename File-Alias Wildcard Ciphering
    - One-Filename File-Alias Wildcard Ciphering w/Forced Alias
    - One-Filename File-Alias Wildcard Ciphering w/Forced Date
    - One-Filename File-Alias Wildcard Ciphering w/Forced Current Date
    - Internet Size Limit Ciphering
- Penknife and Business
  - Ciphers and Keys
  - Business Cipher Management
  - Secure Message Archives
- Penknife The Product
  - Limits of Ciphering
  - Caution: Penknife is a serious cipher. There is no "backdoor."
  - Licensing
  - Disclaimer
  - Author, Company, Publications

Start Up

Overview

What Is Penknife?

Penknife is an easy-to-use data security program for MS-DOS which works well under Microsoft Windows. Penknife protects the information in a file from unauthorized viewing. Protected files can be sent by electronic mail, archived off-site, or just saved locally, hiding the information in those files from irresponsible individuals.

What Does Penknife Do?

Penknife translates (enciphers) files of any sort (word processing files, spreadsheets, programs, graphics, compressed archive files, etc.) into lines of confused-looking text characters. This "ciphertext" contains all of the original information, but keeps it hidden. Unlike many other encryption products, Penknife's ciphertext is real text, and can be handled like ordinary text: Penknife ciphertext can be sent as electronic mail, included in a document, viewed, or even edited. The original file is recovered by "deciphering" the ciphertext file under the correct key.

How Does Penknife Do It?

Penknife uses new, patented Dynamic Substitution technology to implement an especially fast and strong cipher. A user-selected Key Phrase defines the enciphering transformation, and EXACTLY THE SAME KEY PHRASE MUST BE USED TO RECOVER THE ORIGINAL FILE. Penknife has unique key-management facilities to make this easy.

When deciphering, Penknife translates the ciphertext back into exactly the same data as in the original file -- provided the deciphering key phrase is exactly correct. A CRC error-check warns if an incorrect key phrase was used or if the ciphertext file was damaged or tampered with.

Who Should Use Penknife?

Penknife is great for those who want practical privacy for their electronic mail. Most electronic mail has no protection at all, and any number of people can read or even modify it in transit. Penknife makes surreptitious viewing or alteration virtually impossible. Penknife can also be used to protect off-site archives from inappropriate access, provide local privacy, or just to send large files through the Internet with strong error-checking.

Penknife's key-management facilities make enciphering quick and easy for individuals, and allow companies to institute a corporate-wide key management and control policy. A key policy is important, because employers need access to business information which their employees generate.

Everyone who saves enciphered messages and files will appreciate Penknife's support for archived messages enciphered under old keys.

What Do I Need To Use Penknife?

Penknife runs on any system which runs MS-DOS programs, from old 8088 systems through modern Pentium systems, including lap-tops, hand-held DOS computers and even emulated DOS on Unix and other systems. While Penknife's assembly- language code is optimized for modern processors, it is also very efficient on older machines. Unlike many other encryption products, Penknife does not require a numeric co-processor.

The Penknife Versions

Currently, there are four different versions of Penknife: Advanced (the usual choice), Commercial Demo (for evaluation), Decipher-Only (for CD-ROM protection), and Corporate (which limits use to specific keys). Each version has a particular combination of features; only the Advanced version has all Penknife features.

Advanced

The Advanced version of Penknife (typically PENADV.EXE) includes all features.

Commercial Demo

The Commercial Demo (typically PENDEMO.EXE) provides basic cipher capabilities in a small package. The commercial demo does not include alias files (/a, /f), batch files (/b, /k), key-generation (/g), wildcard support, or tree-mode (/t) mass ciphering. However, the Commercial Demo (and all documentation) can be sent -- *without obligation* -- to anyone with whom you would like to communicate.

Individuals can use the demo version until they decide to license the Advanced version. Businesses normally have a 30-day demo evaluation period, but would be better off to contact us for an Advanced demo with more of the features they will need.

Decipher-Only

The Decipher-Only version (typically PENDEC.EXE) has most features, but does not have routines for enciphering (/e) or key-generation (/g). The enciphering routines are physically not present anywhere in the program. This means that -- in certain cases -- the Penknife Decipher-Only version might be exportable.

Corporate

The Corporate version (typically PENCORP.EXE) has most features, but is intended to limit ciphering to only approved keys, and has no key-generation (/g) of its own. This of course means that some entity must produce and distribute alias files containing the approved keys. This gives corporations the ability to define the keys which will be used on corporate data.

Users of the Corporate version can cipher data only using the keys present in their alias file. These users cannot cipher data with a key entered from the keyboard, and so *must* use an alias option (/a or /f).

Corporate version users have no automatic way to make a new alias file. They also normally cannot decipher the alias file, and cannot encipher a new alias file under the alias key. This provides improved corporate control over secret keys.

Installation

All Penknife versions are DOS programs which include internal on-line help panels and a command-line editing facility (for use in Microsoft Windows). Each program is a single file: No separate DLL's or help files are used.

DOS installation

The version of Penknife you have (or want to use) should be copied to some directory listed in your command "path." (To see your current path under DOS, just type "set" -- plus enter -- and look for "PATH.") This will allow the program to be found, loaded into memory and executed. The examples generally assume you have Advanced Penknife, and have renamed the file as PENKNIFE.

Microsoft Windows Installation

A Microsoft Windows installation requires that the version of Penknife you want to use be copied into a convenient directory, perhaps one you normally use for e-mail. The PENKNIFE.PIF file should be copied to the same directory.

Use Program Manager to open the Accessories group and use *File*, *New* to create a new Program Item. Enter the *Description* "Penknife Cipher" and set *Command Line* as the full path to the program (e.g., "c:\pen\penknife.exe). Set *Working Directory* to where you want files without full paths to end up (e.g., "c:\pen). (Eventually, you may put your alias file in the Working Directory.) Enter "p" for *Shortcut Key* (which becomes Ctrl+Alt+P), and check *Run Minimized*. Use Change Icon and select the pocketknife icon, if desired.

After setup, Penknife can be started by double-clicking on the selected icon in Accessories, and then activated when desired with Ctrl-Alt-P, or Ctrl-Esc and *Switch To*. Or use File Manager and double-click on the Penknife program-file. Or use Program Manager *File*, *Run*.

What to Expect

When started, Penknife immediately displays an identification screen. Execution without a parameter starts-up the interaction menu, which can open the help system. If a parameter is found but the source (*FromFile*) is not found, Penknife will stop with an error; otherwise Penknife will request a key, then display the *FromFile* and the destination (*ToFile*) as ciphering starts. On a 25 MHz 386, a 30K file should finish in a couple of seconds.

Simple Operation and Help

Help

Execution without a parameter:

```
penknife
```

will bring up an interaction menu. One option is "Help," and almost all the information needed to run Penknife is available in the help system. Another option is "Enter new command line parameters," which allows Penknife to be used interactively. (Since the program is already running, do not use a "program name" like "penknife" on the interactive command line!)

Simple Enciphering

Basically,

```
            penknife  readme.txt  readme.pen  /e
               ^           ^           ^       ^
               |           |           |       |
     program name   input file   output file   option
```

enciphers the file README.TXT into a new file README.PEN after the program asks the operator to enter a User Key twice. The resulting file can then be sent as a secure e-mail message. (Not available in the Decipher-Only version, and the Corporate version will require an alias; see Using a Key-Alias.)

Simple Deciphering

Similarly,

```
           penknife  readme.pen  readme.res  /d
              ^           ^           ^       ^
              |           |           |       |
     program name   input file   output file   option
```

deciphers the enciphered file README.PEN into the file README.RES. It is normally unnecessary to "clean up" received messages; almost any e-mail header or signature or other text in the enciphered file is skipped automatically in the decipher process. (The Corporate version will require an alias.)

Sending Large Files by E-mail

For transmission on the Internet, a large file can be broken into a sequence of smaller ciphertext files automatically using the /L option:

```
     penknife  big.txt  enc.001  /l  /e
               ^        ^         ^   ^ encipher
               |        |         |
     large file to      |     limit output file size
        encipher        |
                     first ciphertext file
```

This will generate a sequence of enciphered files named ENC.001, ENC.002, ENC.003, etc. until done. The user should send each of the resulting files as a separate message, and include the sequence number of the file in the subject line. (For the Advanced version, and the Corporate version will require an alias.)

Recovering Large Files from E-mail

If you receive a multi-message file, it is easy to accumulate and decipher the messages to get the original file using the append operator:

```
     penknife  enc.001+enc.002+enc.003  big.txt  /d
               ^      ^     ^               ^     ^ decipher
               |      |     |   ...         |
       first msg   append   second msg      result
```

The headers and signatures in each of the messages generally DO NOT have to be removed. There can be no spaces around the "+" append operator. (The Corporate version will require a proper alias.)

Alternately, the messages may be appended as ciphertext (using a text editor or the DOS "copy" command's append mode), and then deciphered as a single large file. Or the individual messages could be deciphered separately and then appended as plaintext.

Creating a Key-Alias

Users of Advanced Penknife should create a file which holds user keys under cipher, so the user need remember only the one key (or pass-phrase) for the key file. We call this an *alias file*, and it can be created or extended by using the /generate option (Advanced version only):

```
     penknife  /g
                ^
                generate
```

The /generate option starts an interactive system to define an alias entry (effective date, alias tag and key), encipher it (the user enters the alias key), and place it at the top of an alias file. If the user does not enter a specific key for the alias entry, a random key will be created.

In addition, the same key can be given a different alias tag (perhaps your own e-mail name) and placed in another file (under another key) for secure transport. The transport key will be installed at the far end, and the different alias tag will be used there to refer to the key used to communicate with you. Normally, each alias tag will in some way refer to the person or entity at the other end.

Ideally, the transport file would be copied to a floppy disk, carried to the far end, and the transport key delivered by separate channel (for example, by postal mail or phone).

Installing a Transported Key-Alias

When an Advanced Penknife alias transport file and its key have arrived, the new alias line can be deciphered with the transport key and immediately re-enciphered under the local alias-file key:

```
     copy  jerrys.pen  temp.tmp
     penknife  /d  jerrys.pen    (enter the transport key)
     penknife  /e  jerrys.pen    (enter your alias key)
```

The resulting ciphertext (now enciphered under the local alias-file key) can simply be placed at the top of the alias file, using a text editor or the DOS "copy" command's append mode:

```
     copy  penknife.mgt  penmgt.old
     copy  jerrys.pen+penmgt.old  penknife.mgt
```

This is intended to save the current alias file, then to collect the ciphertext from the transport file and the saved alias file into a new alias file. Note that the new alias is appended to the alias file *as ciphertext*, and *the existing alias file need not be deciphered*. The saved alias file can then be erased, if desired.

A similar operation is available in the Corporate version: The corporate key-generation facility can provide incremental upgrades in the alias key (or pass-phrase) for each user. The new block of alias entries then is just copied to the top of the existing alias file: No ciphering is needed to add a key to the alias file.

Using a Key-Alias

Once a Penknife alias file is set up, the /a "tag" (a single word) selects an alias in that file:

```
     penknife  file.txt  *.pen  /a fred  /e
               ^           ^     ^    ^   ^encipher
               |           |     |    |
      file to be   resulting     |    the alias to select
      enciphered   file is:      |
                "FILE.PEN"     alias mode
```

Note that fred's key may be updated occasionally, but it is no longer necessary to remember any of those keys. Keys may change, but the alias tag "fred" need not, and also need not be secret. The user only need remember the one secret key to the alias file. In fact, an alias file which has keys effective at future dates provides an automatic and mostly invisible way to institute periodic key changes.

Alias files greatly simplify cipher use. Note that a new alias key can be transported on disk and activated (by releasing the transport key) only after it arrives safely. An alias file could be kept on a floppy and used by making that floppy the "current drive." For ultimate security, that floppy could be kept on the user's person when the cipher is not in use.

OPTIONS, Commands and Features

Command-Line Options

In most cases, a command line will consist of one or two file names plus various options. Each option is specified with a forward-slash "/" plus a letter (or a word with the same initial letter, for better batch-file documentation). Most options (except /k) can be placed anywhere on the command line, in any order, although /a, /k and /m require a data field to immediately follow. The first non-option character-sequence is taken to be the "source" or *FromFile*; the second non-option sequence (if any) the "destination" or *ToFile*.

/Alias <tag> or **/a <tag>**

Use an alias to select a secret key. (Advanced, Decipher-Only and Corporate versions.)

Decipher the closest alias file (in memory only) and scan it for a match to the specified one-word . If such an entry is found, use the associated key-phrase for ciphering. The intent is to select secret keys using non-secret alias tags. Alias tags are not case-sensitive, and the field must immediately follow the /a option.

If environment variable PENMGT exists, that text value is taken to be the full path, file name and type of the local alias file. Otherwise, the program searches for the file PENKNIFE.MGT, first in the current directory, then in the parent directory, then in the root directory on the current drive, then in root on drive C. This allows the automatic use of multiple alias files depending on the current directory.

/a sets the alias tag, the name to search for in an alias file, and the tag itself must immediately follow /a. /a can be used alone or to overrule the tag from /f. Either /a or /f is required in the Corporate version. (Also see Alias Files.)

/Batch or **/b**

Initiate Penknife Batch mode: Decipher *FromFile* in memory and use each plaintext line as a Penknife command line. (Advanced, Decipher-Only and Corporate versions.)

With minor exceptions, each line operates just like it would if typed in at the top level. Option /b is not supported in batch files (no batch files of batch files), and option /k is supported ONLY in batch files, and ONLY as the LAST option on the command line (see /k). Also, options /q and /7 only work in a top level command-line (to DOS). (Also see Batch Mode.)

/Decipher or **/d**

Decipher mode: Decipher Penknife ciphertext. (All versions.)

Normally, e-mail ciphertext need not be "pre-processed" or "cleaned up" before deciphering; headers, signatures and plaintext are usually skipped automatically (see /p).

If the ciphertext file was damaged, or the wrong key used, a CRC error will be reported and ciphering will stop after the current source or *FromFile* file completes. In one-filename decipher mode, the original *FromFile* data will be in the temporary file shown on the screen (usually PEN$$$$$.TMP). The temp file will be overwritten by the next one-filename ciphering, so if the original data are important, they should be copied back to the original file immediately. This recovery cannot be automatic, because if the correct key was used, the deciphered data may be the best we can do. See "One-Filename Mode." One of /d, /e, /n or /g is necessary for operation.

/Encipher or **/e**

Encipher mode: Encipher any file containing any type of data into a file of binary ciphertext. (Advanced, Commercial Demo and Corporate versions.) This text can then be transported as the body of an e-mail message. In one-filename mode, the existing source or *FromFile* will be overwritten with ciphertext and destroyed. It is thus vital that the User Key be entered CORRECTLY in one-filename enciphering, so that the file can later be deciphered. Use an alias file if at all possible, since a wrong alias will simply halt the program without changing the *FromFile*. Also see "One-Filename Mode." One of /d, /e, /n or /g is necessary for operation.

/File-alias or **/f**

Use the file name to create an alias to select a key for ciphering. (Advanced, Decipher-Only and Corporate versions.)

Create an alias tag from the first contiguous alphabetic characters in the file name. Create a message date from the operating system date for that file (decipher) or the current date (encipher). Use that tag and date to search the alias file for the appropriate User Key.

This process can occur dynamically (potentially selecting a different User Key for each file) as the result of wildcard operations. The intent is to support the archive storage of messages in their original enciphered form in an environment where User Keys are changed periodically. Options /a and /m can be used to modify the /f alias tag or date and access other or past or future User Keys.

/Generate or **/g**

Start interactive key generation. (Advanced version only.)

Get information from user, generate a new alias entry, place it at the top of the closest alias file, then place the same key in another enciphered file for transport. (Also see Creating A Key-Alias.)

/Key <keycharacters> or **/k <keycharacters>**

In a Penknife batch command ONLY, set a secret key. (Advanced, Decipher-Only and Corporate versions.)

A batch-command key applies to the alias file, or (absent /f, /a, or /m) the data file. /k is supported ONLY in Penknife batch files, AND ONLY as the LAST OPTION on the line: After /k, the rest of the command line is assumed to be the key. (Potentially, <keycharacters> could include option sequences, such as "/a/b/g/k" etc.)

/Limit or **/l**

When enciphering, limit ciphertext blocks to under 50,000 bytes, a size easily transported on the Internet. This is done by closing the current file, incrementing the filename, and opening a new file. (Advanced and Corporate versions.)

The resulting files should be sent as separate messages; these can be collected at the far end in the correct order by using the Penknife append operator, a text editor, or the append mode of the DOS copy command. Has no effect when deciphering. (Also see Sending Large Files by E-mail and Internet Size Limits.)

/Msgdate <date> or **/m <date>**

Set the date to be used when scanning for an alias (overrule the DOS date). (Advanced, Decipher-Only and Corporate versions.)

Can be used with /a to select a past or future User Key. Can be used with /f to overrule the date from the *FromFile*. Note that the <date> field must immediately follow the /m option.

/m 0 means use the DOS date. Otherwise uses either of two date formats: yy-mm-dd, and mm/dd/yy as selected by the use of "-" or "/" (the numeric fields may have one to four digits).

/Nocipher or **/n**

Disable ciphering. Used to display wildcard file scans and directory-tree scans without affecting any files. (Advanced, Decipher-Only and Corporate versions.)

Display the From and To filenames as they would occur if ciphering were enabled. One of /d, /e, /n, or /g is necessary or the interactive command line is activated.

/Passthrough or **/p**

When deciphering, pass any plaintext found in the ciphertext through to the output file. (All versions.)

Typically used to automatically collect e-mail headers and .sigs (signatures) together with the deciphered message. Has no effect when enciphering.

/Quiet or **/q**

Disable screen output (unless there are errors). Also disable the key-input-prompt beep. (All versions.)

For the original top-level command-line (to DOS) only.

/Tree or **/t**

Scan *FromFile* directory AND ALL SUB-DIRECTORIES for *FromFile* filename matches. (Advanced, Decipher-Only and Corporate versions.)

Normally used with a *FromFile* wildcard expression. BE SURE TO FIRST USE /n WITH /t to display the names of the files which will be ciphered. (Also see Directory-Tree Scanning.)

/Zmode or **/z**

Delete ^Z (Control-Z, hex 1a, display image right-arrow) if that code exists as the last character in a deciphered block. (All versions.)

^Z is the way old CP/M and early DOS editors indicated the end of a text file (EOF). When some editors (and especially the DOS "type" command) see a ^Z they stop reading the file, even though much data may remain (and can be displayed with "`copy con /b`"). Zmode is useful when one or more ciphertext blocks in a file include a ^Z which could hide subsequent blocks and text.

/7bit or **/7**

Display panels as 7-bit ASCII; disable color changes and substitute * for line-draw characters. (All versions.)

Useful when running under emulated DOS on a workstation. For the original top-level command-line (to DOS) only.

Commands for Common Operations

Encipher File "in place"

```
penknife file1.txt /e
```

(Advanced and Commercial Demo versions.) One-filename simple ciphering: Ciphers through a temporary file and places the result back in FILE1.TXT. This hides the original plaintext, but also destroys it, so it is no longer available. The user enters a User Key phrase, twice.

Decipher File "in place"

```
penknife file1.txt /d
```

(Advanced, Commercial Demo and Decipher-Only versions.) One-filename simple ciphering: Ciphers through a temporary file and places the result back in FILE1.TXT. This destroys the ciphertext, so it is no longer available. The user enters a User Key phrase, twice.

Generate Alias Key Effective Jan. 1, 2001

```
penknife  /g  /m 2001-1-1
```

(Advanced version only.) The user enters the alias-file key phrase, the local alias, the far-end alias, the transport filename and key phrase interactively. The most-future keys should always be at the top of the alias file; since /g always places a new key at the top of the file, the user should create new keys in order of increasing date.

Wildcard Alias-Encipher .TXT Files Into .CLO Files

```
penknife  *.txt  *.pen  /e  /a fred
```

(Advanced and Corporate versions.) Two-filename alias ciphering with wildcard file search: Each file matching *.TXT is enciphered into *.PEN. The "closest" alias file is searched for alias "fred" (and the key in effect as of the current DOS date) to find the associated User Key. The user enters the alias-file key phrase, once.

Decipher File Using Alias "fred"

```
penknife  file1.pen  *.res  /d  /a fred
```

(Advanced, Decipher-Only and Corporate versions.) Two-filename alias ciphering: FILE1.PEN is deciphered into FILE1.RES. The "closest" alias file is searched for alias "fred" (and the key in effect as of the current DOS date) to find the associated User Key. The user enters the alias-file key phrase, once.

Decipher In-Place and Pass-Through Plaintext

```
penknife  file1.txt  /d  /a fred  /p
```

(Advanced, Decipher-Only and Corporate versions.) One-filename alias ciphering: This overwrites the ciphertext, which is then unavailable. The user enters the alias-file key phrase, once. The "passthrough" option keeps header and signature lines with the deciphered body of each e-mail message.

Encipher Using Alias Key Active on Dec. 15, 1993

```
penknife  file1.txt  /e  /a fred  /m 93-12-15
```

(Advanced and Corporate versions only.) One-filename alias ciphering: This overwrites the original plaintext, which is then unavailable. The "closest" alias file is searched for alias "fred" (and the key in effect as of 1993-12-15) to find the associated User Key. The user enters the alias-file key phrase, once. If the alias-file key is mistyped, the alias will not be found, and FILE1.TXT will not be damaged.

Decipher Using Alias Key Active on Dec. 15, 1993

```
penknife  file1.pen  *.res  /d  /a fred  /m 93-12-15
```

(Advanced, Decipher-Only and Corporate versions.) Two-filename alias ciphering: Normally, the current date is used in alias searches; /m overrides that with an explicit date.

Encipher a File For File-Alias Deciphering

```
penknife  file1.txt  fred1.pen  /e  /a fred
```

(Advanced and Corporate versions only.) Two-filename file-alias ciphering: Note that the first part of *ToFile* is the same as the alias; this allows the alias to be recovered from the filename. This command enciphers the file under the key for alias "fred" as of the current date. The resulting file FRED1.PEN will produce alias "fred" under /f deciphering, and the file date will hold the enciphering date for the decipher alias search.

Wildcard File-Alias Deciphering

```
penknife  *.pen  /f  /d
```

(Advanced, Decipher-Only and Corporate versions.) One-filename file-alias ciphering with wildcard file search: Find all files in the current directory with type field ".pen", create an alias from the first contiguous alphabetic characters in each filename, create an alias date from the file date for that file, search the alias file for the correct key and decipher each file. In each case the resulting plaintext overwrites the ciphertext file; if one of the files deciphers incorrectly, ciphering stops and the original ciphertext for that file remains in the temp file displayed on the screen.

Note that /f assumes that all *.PEN files have names which convert to valid alias tags, and that each retains the original enciphering date. Even though many different keys may be required for the selected files, the user need only remember and enter the single key for the alias file.

Wildcard File-Alias Deciphering with Alias "fred"

```
penknife  *.pen  /f  /d  /a fred
```

(Advanced, Decipher-Only and Corporate versions.) One-filename "filealias" ciphering with wildcard file search: /a overrides the automatic generation of the alias and just uses the file date associated with each file to select the correct User Key for deciphering (since keys may have changed over time). This assumes that each file retains the original enciphering date. Ideal for ciphertext archives where a single subdirectory contains messages from a single alias.

Change The Alias-File Key

```
penknife  penknife.mgt  /d
penknife  penknife.mgt  /e
```

(Advanced and Commercial Demo versions only.) One-filename simple ciphering: Carefully use one-filename mode to decipher the alias file under its old key, and then carefully re-encipher the file under the new key. Note that there may be multiple alias files in various directories, or the alias file may have a different name if there is a PENMGT environment variable (see the section "Alias Files").

Encipher Multiple Files into Ciphertext Archive

```
penknife  *.txt  arch1.pen  /e  /a fred
```

(Advanced and Corporate versions only.) Two-filename append ciphering with wildcard file search. Each *.TXT filename match causes the associated file to be enciphered into a ciphertext block. Two blank lines are placed between blocks, and all blocks accumulate into ARCH1.PEN.

Encipher Particular Files into Ciphertext Archive

```
penknife  file1.txt+file2.txt+file3.txt  arch1.pen  /e
```

(Advanced and Commercial Demo versions; the Corporate version would need an alias.) Two-filename append ciphering (although *FromFile* steps through three explicit filenames). Each file is enciphered into a ciphertext block. Two blank lines are placed between blocks, and all blocks accumulate into ARCH1.PEN.

Decipher Ciphertext Archive

```
penknife  arch1.pen  *.res  /d
```

(Advanced and Commercial Demo versions; the Corporate version would need an alias.) Two-filename append ciphering. Each ciphertext block in ARCH1.PEN is deciphered and the plaintext accumulated into the common file ARCH1.RES. Each block is announced with either a "." (no data error) or a "*" (data errors found).

Penknife Features

- **New Technology:** Penknife is based on original Dynamic Substitution technology invented by this author, technology which provides an improved tradeoff between speed and security in a software stream cipher. Because the technology is patented, a detailed technical description of the cipher design is available and has been published and discussed in the Usenet News sci.crypt group.
- **Error-Resilient:** Penknife was designed specifically for communications and e-mail, and was designed to be error-resilient: If a transmission error occurs, Penknife can recover at the start of the next ciphertext line; other ciphers may trash the rest of the file.
- **Random Line Keys:** Each and every ciphertext line has its own random line key, in addition to the overall User Key. This greatly improves cipher strength in an error-resilient design.
- **Penknife Ciphertext is Real Text:** Because Penknife produces lines of randomized ASCII characters, the ciphertext can be imported into ordinary text files. This provides the unusual capability of transporting one or more binary files in an ordinary text file, perhaps with surrounding text. Programs or graphics can be transported in this way.
- **Penknife Ciphertext is Only Text:** By avoiding the temptation to have a binary output mode, Penknife avoids the need for users to guess about which mode to use when deciphering.
- **Automatically Handles DOS or Unix Text Lines:** Penknife enciphers files into DOS-style text lines (ending in CR LF), but can decipher either DOS-style or Unix-style lines (ending in LF), automatically and invisibly. Since line-terminators can change in transmission, this can help to reduce confusion, especially for non-technical users.
- **Automatically Skips E-mail Headers or Signatures:** In most cases, Penknife deciphering will automatically skip e-mail headers or signature text. It is normally unnecessary to "clean up" e-mail messages for deciphering.
- **Optionally Passes-Through E-mail Headers or Signatures:** Instead of just skipping e-mail headers and signatures, that text can automatically be "passed through" to the output file. This will keep headers and signatures with the deciphered body of each message. (The /p option.)
- **Checks for Data Error:** Penknife deposits an error-check value in every ciphertext block. This 32-bit cyclic redundancy check (CRC) is also computed when each block is deciphered; if the values do not match, an error is indicated. The error-check supports the enciphering of programs, which should not be executed if they have been damaged in storage or transit. (On the other hand, an ordinary text file might be mostly readable even with a bad CRC because Penknife is error-resilient.)
- **Indicates Use of Wrong Key When Deciphering:** The error-check CRC also detects the use of the wrong deciphering key, a common error in a cryptographic system.
- **Optionally Overwrites Plaintext:** When Penknife is used in "one filename" mode (that is, not ciphering from one file to a new file), the program overwrites the plaintext file, thus (providing DOS cooperates) hiding the original data even from file-recovery programs. No low-level file access table (FAT) disk operations are needed or used.
- **May Support a Secure Delete:** Penknife might be used as the basis of a secure file "delete" operation. Normally, a file "delete" is not secure because the information in the file remains on the disk (until it is overwritten by another file). But to the extent that Penknife "one filename" enciphering overwrites a file with ciphertext, it can be used to "pre-treat" files to make the information on the disk secure so those files can be deleted normally.
- **No Ciphertext Bracketing Required** Penknife ciphertext has no structure apart from text lines. There is no announcement that the body is ciphertext (e.g., BEGIN CIPHER), nor any description of what particular cipher is in use. The recipient is expected to know that a cipher has been applied, and which cipher in particular, along with the secret key.
- **Straightforward Secret-Key Cipher:** Penknife is a "conventional" or "secret key" cipher, and so corresponds to the essence of secrecy: The need for conveying a secret key reminds us that privacy is not available if the other end cannot be trusted. A secret-key cipher does not require "certification authorities" or "trust" with respect to the key. A secret-key cipher automatically "authenticates" a person who possesses the key. A secret-key cipher is not subject to strange "man in the middle" or "spoofing" attacks like some public-key ciphers.
- **Can Break Up Large Files:** When enciphering, advanced Penknife can be commanded to automatically break up a large file into smaller ciphertext files which can be reliably transported on the Internet. When deciphering, either version can decipher messages in a specified order and accumulate them to a common file, thus recovering the original large file. (Alternately, the received files can be appended as ciphertext using DOS "copy" and then deciphered as a single file, or deciphered separately and then appended as plaintext.) (The /L option.)
- **Supports Wildcard File-Search:** Most Penknife versions supports "wildcard" operations, in which multiple files can be selected with a single *FromFile* expression using the DOS wildcard characters "*" and "?". This allows Penknife to encipher or decipher many files with a single key-entry.
- **Supports Directory-Tree File-Search:** Most Penknife versions support directory-tree ciphering, in which the current directory, and all of its sub-directories (and sub-sub-directories, etc.) are searched for a match to the given filename or wildcard expression. This mode supports massive ciphering, such as may be needed when changing keys or ciphers. (The /t option.)
- **Supports Multi-File Archives:** Penknife supports "append mode" operation, where multiple "from" files (selected by a wildcard expression) are ciphered to a single "to" file. Enciphering will produce an archive of ciphertext blocks (which will decipher into a single file). Deciphering will accumulate a plaintext archive from multiple ciphertext files, each of which may have multiple ciphertext blocks.
- **Supports Explicit File-Append:** Penknife supports a form of append-mode operation in which a list of specific files, separated by "+" (with no spaces around the +) is presented as the *FromFile*. This allows explicitly ordering the multiple blocks of a large file sent on the Internet.
- **Supports Explicit File-Sequence:** Penknife supports operation on a list of files separated by "," and presented as the *FromFile*. This allows explicit specification of filenames which are not related and for which wildcard operations will not help.
- **Supports Key-Alias Files:** Most Penknife versions support enciphered alias files, which greatly simplify key management. A simple public alias can select a long secret key from the alias file. Alias files reduce the irritation and the potential for serious error which can result from typing-in long secret keys (twice). (The /a option.)
- **Supports Key Update:** Penknife alias files support "dated" aliases: Each alias has an "key effective date" to indicate when the associated key was (or will be) placed in service. This allows the current key for an alias to be changed to a new key, and periodic key-change is a normal requirement in serious cryptosystems.
- **Supports Access to Old Keys:** Normally, the current date is used to select the appropriate key automatically, but the user can access past (or future) keys by specifying a past (or future) date on the command line. (The /m option.)
- **Supports Archived Ciphertext:** Since old keys can remain accessible, the original ciphertext can be archived without problem. Since occasional key changes are inevitable (and required for security), the ability to support access to ciphertext under old keys is a real advantage.
- **Supports Automatic Key-Change:** Because Penknife supports "dated" alias entries, many different keys can be programmed in advance for the same alias, to take effect at future dates. This can make periodic key-changes automatic and virtually painless.
- **Supports Automatic Access using File Name and File Date:** Most Penknife versions support a "file-alias" facility for developing both the alias tag and the appropriate date from the name and date of the selected file. (The alias tag is built from the first contiguous sequence of alphabetic characters in the file name; the date comes from the file date.) This means that many different files under many different keys can be accessed automatically during wildcard ciphering. (The /f option.)
- **Supports Automatic Access to Archived Ciphertext:** By storing ciphertext messages with their alias tag in their filenames (along with a sequence number), both the alias and the ciphering date can be recovered automatically. This allows a user to avoid remembering dates and aliases for ciphertext, and is especially useful for archiving messages under many aliases.
- **Supports Central Key-Management:** Key-management is important to businesses which have a policy that all business files be enciphered only under business keys. Such a policy can be enforced by audits which attempt to decipher business files using the standard business keys. The corporate version cannot cipher data with a key from the keyboard, but instead requires a previously-generated alias file.
- **Generates New Random Keys:** The Penknife Advanced version supports a "generate" option, to create an alias line with a random key, and then place that line at the start of the closest alias file, as well as (under a different key) in a specified file for transport. (The /g option.)
- **Supports Key Transport:** The "generate" option places both the new key and a "far-end alias" in a "transport file" which is enciphered under a "transport key." The key-transport file can be copied to floppy and carried or mailed to the far end. Once the file arrives safely, the transport key can be provided by a separate channel.
- **Supports Multiple Alias Files:** Most Penknife versions support multiple alias files, searched for in this order: environment variable PENMGT, the current directory, the parent directory, root, and root on drive C: This allows each section of the directory tree to have its own alias file. The alias file might be kept on a floppy which can be personally retained.
- **Penknife Batch Files:** Most Penknife versions also support Penknife batch files, which are enciphered files in which each line is a Penknife command line. This allows substantial processing operations to be pre-programmed for production use. (The /b option.)
- **Non-Infringing:** Penknife does not infringe any known patents. Penknife has been in use for years without problems.

Detailed Operation

Two-Filename Mode

Two-filename mode ciphers from one file to another and the first file is not modified. To start ciphering, just enter:

```
penknife FromFile ToFile /e
```

to encipher, or

```
penknife FromFile ToFile /d
```

to decipher.

FromFile and *ToFile* represent your own filenames. The "from" or "source" file must already exist; it is the file you wish to protect. The "to" or "target" file will be the enciphered version of the "from" file, and will replace any existing file with the same name in the same directory. Both *FromFile* and *ToFile* may be as simple as just a name (if the file is in the current directory), or include a drive-letter and/or "path" specification (to locate the file in some other subdirectory). In two-filename mode, the original file is not modified at all. This can be a safety-net to allow recovery in the event of problems, but can also leave sensitive information on the disk. As an alternative, Penknife has "one-filename mode":

One-Filename Mode

Another way to use the program is to enter:

```
penknife FileName /e
```

to encipher in the "one filename" mode, in which the result overwrites the original file. One-filename mode can be more secure than two-filename mode, because a plaintext copy is not left behind unhidden. But it can also be more risky, since the original file will be overwritten and destroyed.

ONE-FILENAME MODE CAN BE DANGEROUS! Because the original file is normally overwritten, its contents will be destroyed and cannot be recovered.

BE VERY CAREFUL WHEN USING ONE-FILENAME MODE. If a wrong User Key is somehow entered in the same wrong way twice (trust me, it happens), the program cannot know this, and -- unless the error can be reproduced -- the data will have been lost.

Use an alias file if at all possible. An error in entering an alias-file key will simply result in the desired alias not being found, preventing ciphering and also preventing any change to the *FromFile*.

CRC Error In One-Filename Deciphering

If the ciphertext file has been damaged, or the wrong key used when deciphering, Penknife will report a CRC error and stop, leaving the original data in the file described on the screen. In one-filename mode, the original data will be in a temp file (usually PEN$$$$$.TMP).

The program has no way to know whether the CRC error resulted from something small like one bad ciphertext line, or something large like the use of the wrong key. Penknife is designed to be error-resilient and the result (even with a bad section) may be the best that can be achieved with the damaged ciphertext, so things are left as they are. The user should examine the deciphered data which is now stored in the original file.

If the problem was the use of the wrong key (indicated by a completely-random deciphered result), the user should copy the ciphertext from the temp file back to the original file, and decipher under the correct key. It is important to recover the data in the temp file before it is destroyed by another one-filename ciphering operation. One other possibility is discussed next:

"Deciphering" a Text File

When deciphering in "one filename" mode, if the user in error attempts to "decipher" a text file which has not been enciphered, is likely that no valid ciphertext will be found, in which case no deciphering will occur. This case is specifically detected, and the original file will not be modified, but the program will stop with a CRC error.

If this happens, the temporary file (usually PEN$$$$$.TMP) will not be erased, and the original plaintext will be in both files. If the user has placed the temp file on a RAM disk (by setting an environment variable, as described below), there is no security issue. However, if the temp file is on the hard drive, this situation has created yet another dangerous -- but in this case, useless -- plaintext file. The temp file should be scrambled by enciphering in "one filename" mode, and then deleted.

Secure Overwrite Depends on DOS

Although one-filename mode does indeed overwrite the actual disk sectors of the original file in most cases, this depends upon non-guaranteed characteristics of DOS which could depend on the particular version of "DOS" actually used. (Presumably, disk-cache programs could have an effect as well.) A user who is worried about this should format a new floppy, copy one text file to the floppy, encipher that file in one-filename mode, and then use a low-level disk utility to check the disk for sectors containing the original text. Normally there are no such sectors. If there are, it may be necessary to use a disk "wipe" utility to erase all unallocated sectors. Other alternatives include enciphering on floppy disks (which can be bulk-erased to remove all hidden plaintext), or restricting all plaintext files to RAM disk or floppy disk. In some cases it may be necessary to check hard-drive operations as well.

Put Temp Files on a RAM Disk

One-filename ciphering can operate faster (and be more secure) if the intermediate file is on "RAM disk," and this can be arranged (provided you have a RAM disk) by setting an environment variable in AUTOEXEC.BAT:

```
set pentmp=[path and filename]
```

In my system, I use:

```
set pentmp=h:\pen$$$$$.tmp
```

because drive H: is a RAM disk on my system.

The User Key

When invoked without an alias command (such as /a or /f), the program will ask you to enter a User Key, twice (so there is no mistake), after which ciphering will occur. The User Key should be long phrase (30 or more characters) which you will remember, modified to be unlike anything in print.

Penknife Key Phrases

A Penknife key-phrase could be as long as 250 characters. Each key-character can be an upper or lower case letter, a number, or any 8-bit value not used for key-entry which DOS will report to the program.

Some codes which may be a problem for DOS include:

```
^C      abort
^N      output to printer
^P      terminal print-echo
^S      suspend output
^X      cancel
```

Note that ^C means "control-C."

Codes used by Penknife for key-entry include:

```
^[      Esc     erase line / quit
^H      BS      erase last character
^J      LF      ignored
^M      CR      end-of-entry.
^U              alternate erase line / quit
```

Leading and trailing spaces are ignored in key phrases, but internal spaces are significant.

Entering and Editing the User Key

As you enter your User Key it will not appear on the screen so it cannot be seen and stolen. Instead, a sequence of numbers will appear which are the last digit of the number of characters you have entered. While entering your User Key, you may realize that you made a mistake; if so, you can use the backspace key to erase incorrect characters, or hit the Esc key once and start over from the beginning of the line. (Ctrl-U can be used instead of Esc.) Normally, the display digits will be removed, but if your User Key is over about 55 characters, it will wrap to the next line, and display digits will not be erased on a previous line; backspace and Esc will edit the key properly, however.

If you have forgotten the User Key and need to think, just hit the Esc key twice to exit the program.

Some Key Phrase is Always Necessary

When using alias files, the actual User Keys can be random character sequences (since they need not be typed-in or remembered). In fact, the Advanced Penknife /generate option will create random user keys. Still, the alias file itself will need a remembered key, as will any Penknife batch file. Thus, there will always be a need for some keys based on long, rememberable-but-unlike-anything-in-print key phrases.

Alias Files

The ideal way to operate Penknife is to use the alias facility available in most versions. An alias file is an enciphered file which contains the keys for your various uses or contacts. For example, when I want to encipher FileName to "frank" I type:

```
penknife FileName /e /a frank
```

The program then asks for the alias-file key, which need be entered only once. (If the wrong key is given, the alias-tag "frank" will not be found, and the program will stop without damaging data.) So I need to remember only the one key for the alias file, instead of Frank's (and Dave's and Bill's) current secret key (whose actual keys should change routinely anyway).

Creating or Extending an Alias File

The user can create or extend an alias file using the /generate option (in the Advanced version):

```
penknife  /g
           ^
           generate
```

This normally creates a 40-character random new User Key (unless a specific new key is assigned by the user). The new key is collected with a user-supplied alias tag plus a key-effective date (the current date, unless otherwise specified by user) on a single text line. The resulting line is enciphered under an "alias key" from the user, and placed at the start of the closest alias file (if none is found, one is created in the current directory). The same key is also collected with the same date and a different alias tag (for use at the far end) and then enciphered into a "transport file" under a "transport key" from the user.

Dated Aliases

Alias entries include a "key-effective" date to specify when the associated key was (or will be) placed in service. The various alias lines in an alias file should be ordered by date, "future first, past last." Penknife searches for the first occurrence of an alias tag which is active as of the "current" date. The user can set a particular "current" date from the command line (using /m), thus having access to past (or even future) keys. Alias files can be constructed months -- or even years -- in advance, making key-updates painless and automatic.

The Closest Alias File

Penknife seeks an alias file by first checking for an environment variable "penmgt," which it expects to contain the path to and name of an alias file. If the environment variable is not found, Penknife searches for a file named PENKNIFE.MGT, first in the current directory, then in the parent directory, then in the root directory on the current drive, then in root on drive C. This allows the user to have different alias files for different areas of the directory tree structure. We can use the DOS "set" command to establish the environment variable:

```
set penmgt=[path and filename]
```

which will avoid the search, and allow any desired filename. The user might place this command in AUTOEXEC.BAT, where it will be executed automatically when DOS is started. In my system I do not use the environment variable, but instead simply place the working alias file in the root directory on drive C:, thus making it available for use from any current directory. I also have another alias file in my development directory for testing; Penknife uses the testing alias file automatically when I make my development directory "current" (using the DOS "cd" command).

The Transport File

The transport file (generally created when an alias key is generated) must be transported to the far end. There it normally will be deciphered with the transport key and then re-enciphered under that user's alias key. Then the far end user will place the new alias line at the top of their alias file, typically using the append mode of the DOS "copy" command. (See the earlier section "Simple Operation and Help.")

Alias File Format

An alias file is simply an enciphered text file of alias lines. Each alias-file line should contain an "effective" date, an alias "tag," and an associated secret key; each field separated by one or more spaces:

```
2000-12-15 harry          W+dhRbOnz5Ao4Iw07sSjcr5X/dLHm2u24elvx5h
1994-03-24 bob            c7YIvxs8+pfTpk5X3Wqo8Rfs9GvTe1zMPZUmTkE
1993-12-15 fred           EfLoU84fsrN5EwivDK6/6Fpl5qyKGmWEuoHM7Ll
```

Dates are in a form which supports easy sorting: yyyy-mm-dd. Alias tags are single "words" with no internal spaces, do not start with "/",and are not case-sensitive. The secret key starts with the first non-space character after the tag, and includes all characters through the end of the line (except trailing space characters, which are deleted) just like a key typed in on the keyboard. The secret key is case-sensitive and may include internal spaces and any character-codes other than CR or LF. (However, /g produces only the transportable ciphertext characters so that any subsequent editing operations will not have to deal with ASCII control codes or non-ASCII codes). An alias file line cannot exceed about 250 characters in length including, date, tag, secret key and spacing.

Editing Alias Files

Because an alias file is simply an enciphered text file, it may also be edited, either to change alias tags, add new entries which have a language phrase key, or to delete very old keys. However, because an alias file contains perhaps the most sensitive information of any file, effort should be made to minimize overall exposure. For example, a new user-created alias line could be enciphered alone, and then added at the top of the existing alias file as ciphertext (either using a text editor, or the DOS "copy" command's append mode). Two blank or other clearly non-ciphertext lines should separate the new ciphertext from the old ciphertext. Thus, the information in the existing alias file need not be exposed at all.

Date Order Required

Alias files scanned for the first tag-match which is valid as of the "current" date. Thus, alias files should have entries in date order, "future first, past last." All alias lines should have "yyyy-mm-dd" dates so that alias files can be sorted automatically using the DOS "sort" command on the deciphered alias file. (For security reasons, this sort of processing should be done on a RAM disk and the alias-file plaintext should never be displayed on a CRT.)

Penknife Batch Mode

Batch Mechanism

For installations which wish to automate operations, Penknife includes a batch mode. A Penknife batch file is an enciphered file of Penknife command lines. The batch mechanism simply obtains the next text line from such a file, and feeds that line to the normal Penknife system. As much as possible, Penknife processing will look and work exactly the same when processing a true command line, or a command line from a Penknife batch file.

The Differences

There are a couple of differences in batch mode: First, /batch is not recognized in batch files -- it is only used to start batch operations. Next, the /key option is ONLY valid in a batch file. A leading colon (:) on a Penknife batch line indicates a comment. Neither blank lines nor comments are executed, nor are they counted as commands.

Explicit Keys

The /key option (which is only valid at the end of a Penknife batch line) allows batch commands to use alias files (which are enciphered) without separate key entry. Of course, the key for the batch file which starts everything off will have to be entered manually, but only that key, and only once.

Errors Stop the Show

When an error occurs during batch processing, the line-in-error is displayed (with any /key option overwritten) and processing halts, to limit any possible damage to other files. Penknife returns an error-count through the DOS "process return code" which supports special-case processing in DOS batch files.

Other Penknife Features

Append Mode

Penknife has processes which detect first and last ciphertext lines in ordinary text, so it is reasonable to have multiple ciphertext blocks (each enciphered under the same key) in one file. One way to build such a file is to use the Penknife "append" mode: A wildcard *FromFile* expression, with a non-wildcard single file as the *ToFile*. This can encipher multiple files and collect the ciphertext in a single file; each block will be a stand-alone ciphertext segment, with its own CRC. (All ciphertext blocks in such a file will be deciphered into the same plaintext file.)

Similar results can be obtained with an explicit list of files separated by "the append operator" or "+" in *FromFile* only (no spaces are allowed around the +):

```
penknife bob1.txt+john.txt *.pen /e
```

Alternately, the user might use a text editor to manually collect various ciphertext files in a single document, being sure to separate ciphertext blocks with two non-ciphertext lines (either blank lines or text lines with non-ciphertext characters, such as space, punctuation or other symbols). Yet another way would be to use the append mode of the DOS "copy" command to collect separately enciphered files.

When deciphering, CRC results are announced for each block with "." for OK, and "*" for BAD, but processing will continue until the file has been completely processed.

Internet Size Limits

Some parts of the Internet will not handle large messages, so it is wise to break large files down into small and transportable messages. The /L (limit) option will do this automatically:

```
penknife big.fil *.001 /e /l
```

will produce a sequence of ciphertext files: BIG.001, BIG.002, BIG.003, etc. These should be sent as separate messages, with the part number clearly described in the Subject heading.

The process used to step the output file names also supports alphabetic characters. For example:

```
                    ,,,
```

Penknife Quick Start

Ritter Software Engineering
2609 Choctaw Trail, Austin, Texas 78745
(512) 892-0494 ritter@io.com

Contents

- Installing Penknife in DOS
- Alias Files
- Installing Penknife in Microsoft Windows
- Using Penknife in Microsoft Windows

Installing Penknife in DOS

1. **Copy one of the Penknife versions (typically PENKNIFE.EXE, the Advanced version) into one of the program directories in your DOS command "path".** To see your path, just type "set" -- plus enter -- at the DOS prompt, and look for "PATH".

   Alternately, use the DOS "md" command to create a new subdirectory such as "pen" and copy the program there. Then use the DOS "cd" command to move to that directory whenever you use the program. You can use the program from any directory by adding the penknife subdirectory (e.g., ";C:\PEN") to your command path, which is normally set in AUTOEXEC.BAT.
2. **Check out the internal help panels:** Just enter the name of the program (plus return). When the program starts without command-line parameters, it offers an option to enter the help system. Almost all needed information is available in help. Then exit the program.
3. Suppose you have advanced Penknife and a file named FILE.TXT; **to encipher the contents of FILE.TXT into a new file FILE.PEN, enter:**

   ```
   penknife  file.txt  file.pen  /e
       ^        ^         ^       ^ encipher
       |        |      resulting file
       |      source file
   program name
   ```

   Penknife will announce itself, then ask for the User Key, twice. The User Key can be any sequence of characters, but a long unique text phrase is easier to remember than random characters. After getting the key, a 30K file should finish in a second or two. The result is a file of random-looking text lines which can be examined with your text editor or displayed with the DOS type command.
4. **Decipher FILE.PEN to FILE.RES:**

   ```
   penknife  file.pen  *.res  /d
       ^        ^         ^    ^ decipher
       |        |      resulting file (FILE.RES)
       |      source file
   program name
   ```

   Penknife will again ask for the User Key twice, and this must be exactly the same key as was used to encipher the file. (An advanced Penknife alias file makes using keys much easier.)

Alias Files (Advanced Version)

An *alias file* holds a list of hidden keys. A particular hidden key can be selected from the list by using an "alias" or "nickname." The alias file is enciphered to protect the hidden keys, but each particular alias need not be secret. This means that we need only remember the one secret key for the alias file instead of remembering a different secret key for each person.

1. **Create a new alias key:**

   ```
   penknife  newfred.trn  /a fred  /g
   ```

   Here Mary creates a new key to Fred: "fred" is the alias which she will use for the new key. "newfred.trn" is the name Mary gives the file she will later transport to Fred (and she should have previously erased any existing file of that name). Penknife will generate a random key for alias "fred" and automatically place it at the start of Mary's closest alias file, or will create a new alias file.

   Mary will be asked to enter the key for her alias file; this is the most-used and most-important key she will have, and should be a long unique phrase which she can remember.

   Penknife will also place the new random key in NEWFRED.TRN, and Mary will enter "mary" as the far-end tag; this is the alias Fred will use for the new random key, once he installs the key in his own alias file. Mary will also enter another phrase to protect that file during transport.
2. **Encipher or decipher using an alias key:**

   ```
   penknife  file.txt  *.pen  /e  /a fred
   ```

   Now Mary only needs to enter a key once: the key for the alias file. Decipher is similar except using /d instead of /e.
3. **Transport the key to the far end:** Mary copies the transport file onto a floppy and sends it through the postal mail, or gives it to a friend to deliver. She could also use an express delivery service, or just hand the disk to Fred the next time she sees him.
4. **Add a transported key to your alias file** (not in the Corporate version): Fred will need both the transport file and the transport key; then he can decipher that file, immediately re-encipher it under his own alias key, and place the result at the start of his alias file:

   ```
   penknife  newfred.trn  /d
         (then Fred enters the transport key)
   penknife  newfred.trn  /e
         (then Fred enter his alias-file key)
   copy  penknife.mgt  penmgt.old
   copy  newfred.trn+penmgt.old  penknife.mgt
   ```

   Here we assume that Fred is in his alias directory (which is usually the Penknife directory) and so will have direct access to the relevant alias file. Fred might have multiple alias files, in which case the last two commands would be modified and repeated for each alias file where the new alias is needed.

Installing Penknife in Microsoft Windows

1. **Copy the Penknife program to your hard drive.** First create an appropriate directory, such as PEN, using File Manager, double-clicking on root (typically "c:\") and using the *File, Create Directory* selections. Then copy PENKNIFE.EXE and PENKNIFE.PIF from the distribution floppy to that directory, using the File Manager selection *File, Copy*.
2. **Create a new program item under Program Manager, and connect it to an icon.** Use Program Manager to open the Accessories group and use *File, New* to create a new Program Item. Enter the Description "Penknife Cipher" and set *Command Line* as the full path to the program (e.g., "c:\pen\penknife.exe"). Set *Working Directory* to where you want files without full paths to end up (e.g., "c:\pen"). (Your alias file will also normally be in that working directory.) Enter "p" for *Shortcut Key* (which becomes Ctrl+Alt+P), and check *Run Minimized*. Use *Change Icon* and select the pocketknife icon, if desired.
3. **If you do not have PENKNIFE.PIF, construct it!**

   Use the PIF Editor (in Program Manager group Main) selections *File, New* to start a new definition. Then set *Program Filename* as the complete path to Penknife (e.g., "c:\pen\penknife.exe"). *Optional Parameters* and *Start-up Directory* should be empty. *Video Memory* should be "Text". *Memory Requirements* for *KB Required* and *KB Desired* should both be "150"; for both *EMS* and *XMS memory, KB Required* and *KB Limit* should be "0". *Display Usage* should be "Windowed", *Close Window on Exit* checked, and neither *Background* nor *Exclusive Execution* need be checked.

   Of the *Advanced Options, Uses High Memory Area* need not be checked, but *Emulate Text Mode* should be. Then use the PIF Editor selections *File, Save As* to save the new file as PENKNIFE.PIF, in the same directory as Penknife (e.g., "c:\pen\penknife.pif").

   The "PIF" file tells Windows how to set up the DOS window when Penknife is invoked. 4. Now Penknife can be started by double-clicking on the selected icon in Accessories, and then activated when desired with Ctrl-Alt-P, or Ctrl-Esc and Switch To.

Using Penknife in Microsoft Windows

1. **Penknife can be started by double-clicking on its icon** (probably the pocketknife in the Accessories group), or by double-clicking on PENKNIFE.EXE in File Manager. Penknife will then lurk in the background waiting for interactive command-line parameters.
2. **When desired, bring up the waiting Penknife** (use Ctrl-Alt-P or Ctrl-Esc and Switch To). Normally, Penknife waits in the interaction menu where there are three options: "h" to enter the help system, "e" to "Enter new command line parameters", and "q" to end the program. If you enter a command line, Penknife will execute it and then return for another command.
3. **When done, type "q" to quit**, or just minimize the window. Type "q" to quit the program and close the window, or just minimize the window to keep Penknife instantly available.

Enciphering E-Mail

1. **Create the message and save it as a file.** Use an editor, Notepad, or some other application to create the message, and then *File, Save As* to save it. Be sure to remember the directory and filename you give to it (for example, C:\EMAIL\JOHN01.LTR).
2. **Encipher the file with Penknife.** Open up Penknife (use Ctrl-Esc, Switch To), select "Enter" and enter an encipher command (for example, "/e c:\email\john01.ltr"). You will be asked to enter a User Key and then Penknife will encipher that file.

   If you have Advanced Penknife, it is better to establish an alias file to hold your keys. You can use public alias tags to select any of the keys, and you need only enter the one key for the alias file, once. A typical command would be: "/e c:\email\john01.ltr /a johnj" after you establish a key for "johnj".
3. **Send the resulting file as your e-mail message.** One way to do this might be to get into the e-mail application and do *File, Open*. Alternately, some other application could do File, Open; and the Clipboard used to Edit, Cut and then Edit, Paste the message into the e-mail editor.

Deciphering E-Mail

1. **Save the received e-mail message to a file.** Use *File, Save As* in the e-mail application to save the message. Be sure to remember the directory and filename you assign (for example, C:\EMAIL\FRED1.MSG).
2. **Decipher the file with Penknife.** Open up Penknife (use Ctrl-Esc, Switch To), select "Enter" and enter a decipher command (for example, /d c:\email\fred1.msg). You will be asked to enter a User Key and then Penknife will decipher that file.

   Again, if you have Advanced Penknife, it is better to establish an alias file for your keys. Then "/d c:\email\fred1.msg /a fredw" will select the key for "fredw".
3. **Read the deciphered message.** The resulting message can be viewed from almost any editor, Notepad, the e-mail application (usually), or any other application which has File, Open and will show the file.

Terry Ritter, his current address, and his top page.

Last updated: 1995-12-14

The Penknife Cipher Design

An Error-Resilient Stream Cipher for E-mail

Terry Ritter

At the design level, the Penknife program is an application of Dynamic Substitution to an error-resilient stream cipher. At the user level, Penknife is an e-mail cipher program which should work with any e-mail system. The Penknife program supports serious key management, which is the heart of a modern cipher.

Contents

- Overview
- Components
  RNG, Jitterizer, Dynamic Substitution Combiner, Ciphertext Feedback
- Structure
- Operation
  User Interaction, Initialization, Ciphertext Lines, Blocks
- Comments
- Strength Arguments
  Brute Force on RNG, Brute Force on User Key, Chosen Key, Differential Cryptanalysis, Ciphertext Only, Known Plaintext, Chosen Plaintext, Line-Key Security, The Extent of a Break
- Current Implementation

Overview

Penknife builds on the basic concepts of a conventional stream cipher: The classical stream cipher combines a confusion stream with a data stream to produce a ciphertext stream. A conventional stream cipher uses some sort (pseudo-) random number generator (RNG) which is additively combined with data by exclusive-OR. The Penknife design differs by using:

- a random 32-bit value on each ciphertext line,
- a nonlinear isolation component after the RNG,
- a nonlinear combiner with internal state,
- two levels of combining,
- ciphertext fed back into the RNG, and
- a final substitution on the ciphertext.

Penknife ciphertext consists of independent text lines. An error in one line will affect only that line, and not the entire message. Such an error will be detected by an overall CRC.

The Components

The Penknife components include a random number generator (RNG) and Jitterizer in the confusion subsystem, and Dynamic Substitution and exclusive-OR in the combining subsystem. And ciphertext feedback further complicates the RNG subsystem.

The Random Number Generator

The Penknife RNG produces 16-bit values from a 63-bit internal state. The RNG uses both a standard 32-bit Linear Multiplicative Generator (LMG), and a degree-31 Linear Feedback Shift Register (LFSR). (The deg-31 LFSR primitive is a 17-nomial for better randomization effects.) The two RNG elements step separately, with their most-significant 16 bits combined by exclusive-OR to produce the desired RNG output.

The Jitterizer

The Jitterizer is a simple but effective mechanism which deletes values from a linear sequence, and confuses the values which are accepted. First, an RNG value is collected, in which two bits represent the number of RNG values to "skip" (1..4), and six bits represent the number of RNG values to "take" in the next take group (1..64). Then the indicated number of RNG values are skipped, with the last value retained as an "offset," to be exclusive-ORed with all members of the upcoming take group. Then another RNG value is produced, exclusive-ORed with the offset, and returned. Subsequent Jitterizer calls collect and offset each RNG value until the number of take counts expire and it is time to skip more RNG values and produce a new offset. The Jitterizer deletes about 10% of the input sequence, but the sequence skips and changing data offsets make the sequence very nonlinear.

The Combiner

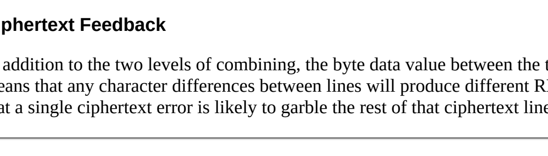
The Penknife combiner subsystem takes a single-byte plaintext input, plus two RNG confusion bytes, and produces a one-byte ciphertext result. This occurs in two sequential combinings: The first combiner is a nonlinear byte-wide Dynamic Substitution combiner [1,2], which adaptively randomizes the input codes. This means that repeated uses of the same character are generally assigned different codes. The output of the Dynamic Substitution combiner is then fed into a linear exclusive-OR combiner, which produces the ciphertext output. (Note that a two-level combiner only makes cryptographic sense when at least one of the combining levels is nonlinear.)

Ciphertext Feedback

In addition to the two levels of combining, the byte data value between the two combiners is fed back into the RNG subsystem by addition mod 2**16 with the least-significant word of the LMG. This means that any character differences between lines will produce different RNG sequences for the rest of each line. This helps make it difficult to explore the Dynamic Substitution tables. It also means that a single ciphertext error is likely to garble the rest of that ciphertext line.

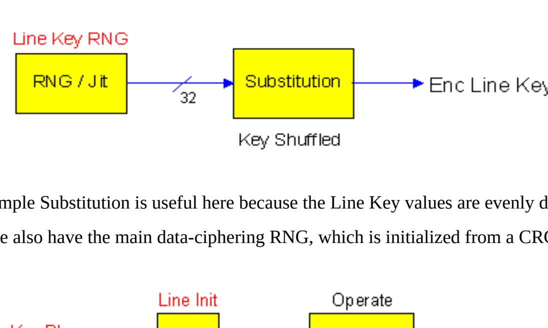
Structure

The Penknife cipher uses two identical isolated RNG's: one to develop the Line Key values, and the other to cipher the data. A substitution table is used to cipher Line Key values.



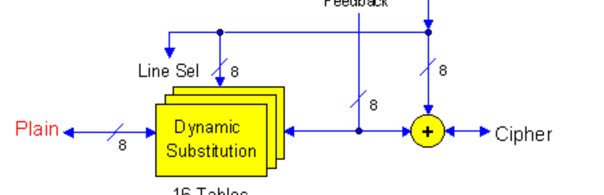
Simple Substitution is useful here because the Line Key values are evenly distributed, thus avoiding the usual attack on Simple Substitution.

We also have the main data-ciphering RNG, which is initialized from a CRC of the key phrase or random key. The main RNG directly drives the combiner.



The data combiner is a two-level structure. The first combining level consists of a particular Dynamic Substitution table selected from an array of sixteen such tables. The selection is made using a value from the data RNG once per ciphertext line. The second combining level is exclusive-OR. The combiner thus takes eight bits of data and sixteen bits of confusion and produces eight bits of ciphertext.

There is an 8-bit to 6-bit converter which takes 53 enciphered data bytes and 4 enciphered Line Key bytes and produces 76 bytes of 6-bit values (0..63).



A keyed substitution table is used to translate the 6-bit values to network-transportable characters. Keying this table is useful because it provides some added strength at essentially no cost. The 6-bit symbol-frequency statistics are flat, which avoids the usual attack on Simple Substitution.

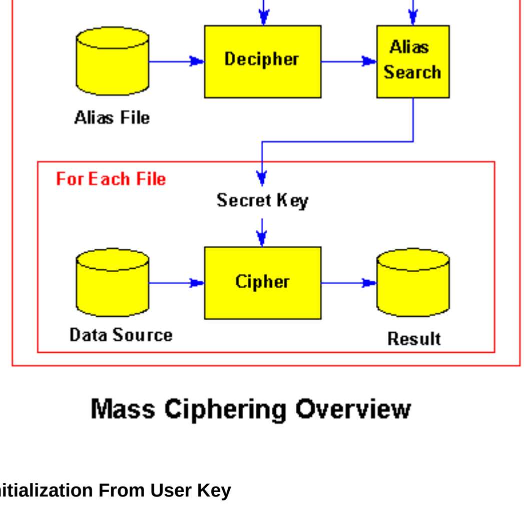
Operation

It is important to keep in mind that the cryptographic cipher proper exists only in the context of a larger design. It seems useful to first generally describe how ciphering is invoked.

User Interaction

Ordinarily, we expect the user to have an alias file of secret keys, each of which is accessible by public alias.

The cipher is given an alias, finds the "closest" alias file, deciphers that file in memory only (using the same Penknife cipher), and produces the current secret key associated with that alias. The "current secret key" can be a long random value, and is the key used for actually ciphering data.



Mass Ciphering Overview

Initialization From User Key

A key of arbitrary length and content is converted into 63 bits of RNG state using Cyclic Redundancy Check (CRC) operations. CRC-32 is used to build the state for the LMG, while a degree-31 primitive builds the state for the LFSR. This 63-bit base state is placed in the line-key RNG, where it is first used to set up various substitution tables. Each table is first initialized with entries in counting sequence, and then shuffled under control of the Jitterized RNG output, to produce particular arbitrary table permutations associated with a particular User Key.

The network-ASCII table is also permuted under the control of the base-state sequence, thus producing a character mapping unique to each User Key. This is an additional layer of protection which is essentially free, since the mapping is required in any case.

The separate line-key RNG is randomized with the current date and time (and -- in the PC environment -- samples from the high-speed timer) before being used to produce line-key values. Each line key is created from two line-key RNG operations, and is enciphered by byte-wide Simple Substitution. Note that this substitution occurs on highly-randomized values; the character-frequency statistics which normally make substitution weak do not exist at this point.

Ciphertext Lines

We assume that the User Key has initialized the RNG's and all the tables.

The 32-bit line key associated with the ciphertext line is treated as an extension to the User Key, being combined into the base RNG state with CRC-32 and a deg-31 CRC. This produces the initial RNG state for the confusion sequence or "running key" for each line.

The resulting RNG state for each line selects one of 16 possible combiner base-states and used as the combiner for that line. The Dynamic Substitution state and RNG state then progress -- change -- through the end of that ciphertext line.

To encipher each ciphertext line, up to 53 plaintext bytes are acquired from the input. Then the Line Key RNG generates a 32-bit Line Key value. The line key value is used to transform the base state of the data RNG into a particular state for that line, and that RNG state selects one of 16 Dynamic Substitution tables.

For each data byte, the data RNG is stepped to produce a 16-bit confusion value. The data byte is combined with confusion first by Dynamic Substitution then exclusive-OR, producing a single enciphered byte. In this way, up to 53 bytes are accumulated.

When the ciphertext line is finished, it is collected with the enciphered line key and translated from 8-bits-per-byte to 6-bits-per-byte format. The 6-bit values are simultaneously coded for transmission and enciphered in a keyed substitution table.

Blocks

Because the Penknife cipher is designed to be error-resilient, it has no external structure. There are no --BEGIN CIPHERTEXT-- or --END CIPHERTEXT-- flags. The ciphertext blocks produced by different files can be concatenated, which is surprisingly useful.

There is an overall CRC-32 of the plaintext data in each block; the CRC result is enciphered as data. When the file is deciphered, overall ciphering correctness is indicated to the user. The CRC-32 field is deleted and the resulting file length is correct to the byte.

Comments

A random overall message key is sorely missed. However, such an entity would mean that a transmission error in the message key would destroy the entire message, contradicting the design goal of error-resilience.

It is interesting to consider the cryptographic use of the much-maligned CRC function. Penknife uses CRC to produce a cryptographic transform from an initial RNG state to some arbitrary RNG state depending on the User Key phrase. (We could consider the result to be a binary vector which is then combined by exclusive-OR with the initial state.) In its favor, CRC is very fast and has a strong mathematical basis for producing a random-like hash from the biased symbol distributions in ordinary text. CRC is linear so it is also weak, in the sense that we could predict the transformation, but that is irrelevant: If The Opponents knew either the external User Key phrase or the internal starting RNG state, they would not have to attack the cipher.

In practice, the 32-bit line keys mean that, for a given User Key, there are 2**32 (4 * 10**9 or 4E9) different line keys. The Birthday Paradox tells us that we can expect (probability 0.5) two identical line keys (thus initial line states) after only 1.18 * 2**16 or 77E3 lines. (With 53 data bytes per line, 77E3 lines would imply 4 MB of data.) However, at this level we still have (prob. 0.5) only two lines with the same initial RNG state, and those states remain similar only until the first character difference between the two lines has been encountered. As more lines become are available, we may find other sets of lines in which two lines have same line key, but it will (probably) take MANY more lines before we can expect to find even three lines enciphered under one key. Clearly, collecting lines enciphered under one User Key and one line key is going to be very expensive.

(Obviously, given enough ciphertext, it will eventually be possible to find many lines with the same line key. This would allow an Opponent to start the sort of technical attack which is discussed later. This is only one of several reasons to change User Keys periodically; another is the perhaps more-realistic possibility that someone has stolen the key without leaving any signs or other indication.)

Even though the RNG has 4E9 different initial line states, there are still only 16 Dynamic Substitution tables under any particular key. It is not clear how one would go about exposing this internal state.

The ASCII-only output is extremely flexible. Because the jumbled ciphertext lines really are text lines, they can be included in text documents as a way to transport or save binary files. Once Penknife is in place, it can be used to decipher its own upgrades sent by e-mail.

The 64-symbol ciphertext means that for every three 8-bit data bytes we get four 6-bit ciphertext bytes; for every 53 data bytes we have 4 line key bytes; and every ciphertext line must have CRLF. Ultimately, 53 data bytes produce 78 ciphertext characters, a data expansion of 47 percent. This is somewhat more expansion than in other e-mail ciphers (which may not be error-resilient) due to the use of line keys in the Penknife design. But any e-mail -oriented cipher will expand data when they are enciphered, and so will be somewhat inefficient for extensive local file ciphering. On the other hand, the lack of a binary output mode means that the enciphering operator does not need to select "binary" versus "text" output, and the deciphering operator need not be confused by the possibility of two incompatible modes.

Penknife assumes that the plaintext is binary data, which is then enciphered with an overall CRC. The mapping to 64-symbol network-ASCII is random, under a particular User Key. The ciphertext contains no control fields: There is no cipher identification, nor indication of ciphering mode. The only Penknife data structure is the ciphertext line, which should be virtually indistinguishable from any other random 76-character line. Deciphering recovers the original data, strips off the CRC, and announces the CRC result.

Because of the ciphertext structure of independent text lines, ciphertext can usefully be concatenated *as ciphertext*, without deciphering. This property is useful for collecting ciphertext archives and also creating files which are updated from time to time. The prime example of this is *alias files*, which hold user keys associated with public aliases. As keys are added and updated, new sections need only be appended to the existing file, without deciphering (and thus exposing) these critical contents.

Strength Arguments

Brute force on the RNG:

> The Opponent tries each possible RNG state until the cipher is broken.

There are 63 bits of RNG state: Clearly, it is "only" necessary to step through all possible states, setting up all the dynamic tables each time, to "break" the cipher. This is the design strength of the cipher.

Brute Force on the User Key:

> The Opponent tries each possible User Key until the message deciphers properly. Try most-likely keys first. (This is most applicable to the alias file key, which may be a language phrase.)

No cipher can do very much about this attack if a user really wants to use a weak key. The advanced Penknife program supports the automatic generation of random keys into an alias file, where they can be selected and used with a non-secret alias tag. This does not completely solve the problem, however, because the alias file itself requires a User Key.

Penknife supports User Keys of virtually arbitrary length, but even a long key is not necessarily "strong." For example, concatenating the names of family members into a long key will not make a good key. A strong key must be unique and unsearchable, and never written down or stored as plaintext in other files. Penknife alone cannot assure this.

Since the alias file User Key must be entered, it can be stolen, and so should be changed periodically. This is easy to do by deciphering the alias file under the current key and then immediately enciphering under the new key. In most cases, it is not the cipher design but instead the alias-file key which is the weakest part of the system.

Chosen Key:

> The Opponent tries various keys, adaptively, and compares the resulting ciphertext to the real ciphertext, to try and gain insight into the correct key value.

Given that CRC is used to generate the internal RNG base state, it is hard to see how any particular bit arrangement could possibly be preferred, or occur unusually often. This is one big reason for using CRC as opposed to, for example, a "cryptographic" hash which has no known strong mathematical basis.

Differential Cryptanalysis

> Here the Opponent exploits properties of particular known substitution tables or transformations. Repeated similar transformations on data may produce an uneven distribution of bit probabilities. This information can be used to effectively reduce the number of "rounds" in an iterated block cipher.

Differential Cryptanalysis does not appear to apply to this cipher, because all Penknife tables are "keyed." That is, all Penknife tables are unbiased arbitrary permutations created by an RNG initialized by a particular key. Since proper shuffling algorithms are used, every table permutation is equally probable. This means that the particular table arrangements will be unknown to The Opponent, so the advantage of prior knowledge of a particular table arrangement is lost.

Ciphertext Only

> The Opponent accumulates a mass of ciphertext material and tries to find relationships within the data which expose successive levels of the mechanism, until the cipher is broken.

The data flowing through the Penknife cipher are extensively randomized, first by adaptive Dynamic Substitution, and then by exclusive-OR with a pseudo-random sequence. The data then pass through 3-to-4 byte conversion and the network-ASCII mapping table (which is also a function of the User Key). By itself Dynamic Substitution removes the usual symbol-frequency statistics to produce a random-like result, and the exclusive-OR also produces a random-like sequence. Consequently, it is hard to see how one could get a statistical hook into even the network-ASCII table, let alone the rest of the cipher.

Known Plaintext

> The Opponent somehow "obtains" some amount of plaintext and the corresponding ciphertext. Ordinarily, this attack is on a conventional stream cipher using exclusive-OR combining, which will immediately reveal the confusion sequence. The confusion sequence and knowledge of the RNG design are then used to develop the original cipher state, which breaks the cipher.

Penknife has a two-level combiner which combines a single byte of data with two bytes of confusion to produce a single byte of ciphertext. Accordingly, a single known-plaintext byte cannot, by itself, describe the two bytes of confusion which produce the ciphertext.

Nor do single bytes reveal the Dynamic Substitution table state, because the selected table element changes at random after use. In contrast to other stream-cipher designs, known-plaintext will not immediately resolve the confusion value so that the RNG can be attacked.

On the other hand, if we could find enough lines which have the same line key (under one User Key), but different initial character values, we could try to describe the initial state of the selected Dynamic Substitution table. Then, given lines with the same initial byte, but every possible second byte, we could try to describe the complete state of the Dynamic Substitution table after a single character. This would allow us to identify which elements of the table have changed, and thus, implicitly identify the hidden first RNG value, on the way to attacking the Jitterizer (and RNG). Of course, such an attack seems to require that every possible plaintext byte occur both as the first and the second (and then the nth) element, all in the same line key. Because line keys are random-like, this would imply a huge amount of known-plaintext.

All this assumes that some method has been found to surmount the network-ASCII table, or to postpone this mapping for resolution in later analysis. It is not clear how one could do this.

Chosen Plaintext

> The Opponent somehow arranges to use the cipher at will, but does not know the key. The cipher is used to encipher potentially huge amounts of data specifically -- and sometimes dynamically -- chosen to reveal the cipher.

(It is not clear that this would be a realistic attack in the context of normal secret-key end-user-cipher operation: If The Opponent has access to the ciphering computer, the compute state probably could be stored to a file, which would then contain the internal key. This is more a "black bag" attack than cryptanalysis, but would far easier and thus far more likely under the stated conditions.)

A chosen-plaintext attack can be no harder than the same attack under known-plaintext; the question is whether it would be any easier. Since the line-key values are not under the control of the user, even controlled ciphering is going to require a huge amount of ciphering simply to explore the first and second table states. If the table states can be exposed, we could acquire the 16-bit result from the Jitterizer. But the RNG contains 63 bits of state, so there are about 2**47 ways in which the RNG could have produced that value. This is searchable to the extent that the 63-bit RNG is searchable. But when we include the effect of the 32-bit state inside the Jitterizer, clearly, any RNG output value can produce any Jitterizer result, depending on the internal Jitterizer offset set earlier in the sequence. So the RNG may be attackable, but seems quite unlikely to be broken with a single Jitterizer value, and collecting Jitterizer values indirectly by exploring table state is very, very expensive.

Exploring table state involves the coincidental appearance of every possible character in the last position of an expanding sequence. Consequently, this approach would seem to be exponentially difficult, and to require exponentially more ciphertext for however many elements are required to break through the Jitterizer into the RNG's themselves.

Worse, the RNG sequences on two lines with the same line key differ after the first different data byte, and the value which makes this difference is hidden between the combiners. Thus, we seem to need the Jitterizer value to expose the hidden value to be able to expose the next Jitterizer value.

The Opponent would no doubt like to be able to simply encipher every possible one-character message under a given key. But, again, the line keys are not under user control, and they are 32 bits wide. Still, if there were a complete single-character enciphering under one line key (and, thus, selecting a particular initial Dynamic Substitution table), it should be possible to completely explore the table (with some constant offset from the exclusive-OR combiner). Then, given any single initial character, if all possible two-character messages could be obtained (under a single line key!), then the second table state could be explored, and the hidden Dynamic Substitution confusion value identified (and this would be absolute, not relative to the exclusive-OR). But all this depends on getting a complete set of ciphertext values under a single line key, and it is not clear how one can do that other than chance. Again, this would be very, very expensive.

And, once again, all this assumes that some method has been found to surmount the network-ASCII table, or to postpone this mapping for resolution in later analysis.

Line-Key Security

Clearly, the line keys provide significant protection, so it is interesting to speculate about the need for line-key randomness or ciphering.

Without line keys, the cipher is immediately subject to (a still expensive) chosen-plaintext attack. This may or may not be a big deal in an end-user cipher, assuming that only the key-holders encipher data.

Without line keys, acquiring known-plaintext with "every" first byte code could explore the Dynamic Substitution table. Acquiring known-plaintext with "every" second byte code (for some particular first byte code) could explore the Dynamic Substitution table again, identifying differences, and indicating a particular initial Dynamic Substitution confusion value, leading to the start of an attack on the Jitterizer/RNG. If "every" second byte code is not available after one particular first byte code, the various Dynamic Substitution tables could be explored, but after the first character, the internal RNG sequences begin to diverge anyway, so to some extent each must be explored separately until the RNG can be cracked, and it is not clear what it would take to do that. Still, this attack would be much easier without line keys.

If the line keys were not enciphered, their effect on the RNG state would be easily computable. Although the original RNG state would not be known, the line-key effect on that state would be known. This would mean that the ultimate change in the first RNG value would also be known. However, the first RNG value on each line is not available for analysis, since it is used by the Jitterizer for "skip" and "take" values. The first RNG value which is produced from the Jitterizer occurs later, and is confused by Jitterizer offset, so the relationship to the line-key is not direct or immediate. And The Opponent would still have the problem of exploring the Dynamic Substitution tables (as well as the Network ASCII table).

If line key values were produced, say, by a counter, they would be difficult to protect by fast, simple cipher. (By itself, a single arbitrary value is easy to protect. However, a sequence of values, produced in a known way, is harder to hide.)

Line key values could be produced by some sort of really-random system. But if really-random stuff is necessary, the design is not likely to be very portable.

Consequently, line keys are now produced by second Jitterized RNG. One does have to wonder whether some characteristic of the RNG could survive both the Jitterizer and Simple Substitution to help The Opponent resolve the line-key RNG state. Resolving that RNG could give insight into line-key effects on the ciphering RNG, and so help crack the overall system. But the use of an RNG which is a combination of two well-known types of basic RNG mechanism, and the nonlinear action of the Jitterizer, make it difficult to imagine how any single RNG characteristic could survive unscathed.

Note that any enhancement to produce line-key values in some stronger random way would be completely compatible with the existing design.

The Extent of a Break

Absent an overall Message Key, and with small (32-bit) line keys, breaking Penknife probably means resolving the initial RNG state, which is the arbitrary value set by the User Key. The cipher thus remains broken until the next key change (which obviously must not be done under the "cover" of the broken cipher). The original, hand-exchanged keys might be used simply to transfer random keys (which are easy to use in enciphered alias files). In this case, the usual periodic key change--which is necessary anyway, since a key may have been exposed without our knowledge--should reset security to original levels as long as the original keys have themselves not been exposed. In the advanced implementation, actual transmission keys can be changed periodically, through the prior establishment of dated aliases for future keys. The alias file User Key should be changed periodically as well.

Current Implementation

The current Penknife implementation is a relatively small (47K including substantial "help" information), relatively fast (80 KBytes/sec on a 486/DX2-50), command-line program for DOS on IBM PC-type machines. The current design includes an interactive command-line entry mode which is easy to use under Microsoft Windows. Inner ciphering loops are written in hand-optimized assembly-language for best speed, but the program will run on an old 8088 machine, and requires no numeric co-processor.

The overall cipher design includes extensive key-management features to help create, transport, and use large random keys. Enciphered alias files allow many secret keys to be kept under a single alias-file key, and then selected by public alias. This supports a single key-entry by the user, since, if the key is entered incorrectly, the alias will not be found and ciphering will not occur.

Aliases inherently support painless key-update, since the user continues to use exactly the same alias for each new key. Because each Penknife line is ciphered separately, new keys can be added to the enciphered alias file without deciphering and possibly exposing the existing keys.

Dated-aliases allow keys to be programmed in advance, then automatically selected and used when the date changes. The ability to access old, replaced keys by date supports access to old messages and entire archives. Alias files also support the centralized key-management which is important for business.

Penknife can handle messages containing both plaintext and ciphertext, can automatically find and decipher the ciphertext, and optionally pass the plaintext through to the output file. Because of this, email files normally do not have to be cleaned up for deciphering, and headers and signatures can be kept with the deciphered text. And all of this is done WITHOUT announcing that the text is ciphered, or the name of the cipher, or using "--BEGIN CIPHERTEXT--" and "--END CIPHERTEXT--" bracketing.

The Penknife cipher is a serious commercial design, and is part of a growing family of serious ciphers and cipher engines for software developers. These designs use a wide range of technology to achieve a wide range of speed versus performance tradeoffs in software implementation.

References

[1] Ritter, T. 1990. Substitution Cipher with Pseudo-Random Shuffling: The Dynamic Substitution Combiner. *Cryptologia*. 14(4): 289-303.

[2] Ritter, T. 1990. Dynamic Substitution Combiner and Extractor. *U.S. Patent 4,979,832*.

[3] Ritter, T. 1991. The Efficient Generation of Cryptographic Confusion Sequences. *Cryptologia*. 15(2): 81-139.

Terry Ritter, his current address, and his top page.

Last updated: 1996-05-31

Cloak2 Features

A strong data security program with key management, for use under Microsoft Windows and DOS.

Contents

- Overview
- Operation
- Examples of Commands
- Features
- Key Management

Overview

Cloak2 takes a secret key phrase and one or more filenames, then transforms or *enciphers* files into random binary data. The original file contents can be recovered only by *deciphering* with exactly the same key. Files thus protected can be saved locally, archived off-site, or sent by e-mail (using conventional binary protocols) without exposing their contents.

Cloak2 enciphered alias files hold secret keys and allow them to be selected by public aliases, thus minimizing the impact of secrecy on ordinary users. Keys can be updated while users continue to use exactly the same alias in exactly the same way. Dated alias allow access to old archived ciphertext protected by outdated keys.

Now available in Advanced, Commercial Demo, Decipher-Only and Corporate versions for DOS and Microsoft Windows. Not for export.

Operation

For greatest ease and security, the user should have an *alias file*. An alias file is an enciphered file of public aliases and related secret keys. The user can thus select one of many secret keys by supplying the key to the alias file plus the public alias for the desired key.

In normal operation, the user supplies the input filename, output filename (if different), the alias, and selects encipher or decipher. The program requests entry of the alias-file key, once. The program then finds the "closest" alias file, deciphers it in memory only, finds the indicated alias, then uses the secret key associated with that alias to cipher data.

The "generate" function in the advanced version automatically creates or adds to a local alias file, and will encipher the new key in a different file for transport. The transport file can be used as a beginning alias file, or added to an existing alias file, without deciphering the alias file.

Examples of Commands

- **Encipher Multiple .TXT Files Using Alias "fred"**

```
cloak2  *.txt  *.clo  /e  /a fred
```

- **Decipher Multiple .CLO Files In-Place Using Alias "fred"**

```
cloak2  *.clo  /d  /a fred
```

- **Decipher Using Key Active on Dec. 15, 1993**

```
cloak2  file1.clo  file1.res  /d  /a fred  /m 93-12-15
```

- **Change Key to Alias File**

```
cloak2  cloak2.mgt  /d    (Enter Old Key)
cloak2  cloak2.mgt  /e    (Enter New Key)
```

- **Encipher Particular Files into Ciphertext Archive**

```
cloak2  file1.txt+file2.txt+file3.txt  arch1.clo  /e
```

Features

- **Easy to use** under Microsoft Windows or DOS.
- **Fast:** Up to 150K bytes/sec on large files (on a 486DX2/50).
- **Extremely Strong:** Uses a 992-bit internal key, with a random 992-bit message key in every file.
- **Small:** Under 50K including on-line help.
- **Enciphers any file of any sort and recovers the original data without loss.** CRC error-detection checks each deciphered file. CRC also detects the use of a wrong deciphering key.
- **No "wrong" operating mode:** produces only binary ciphertext.
- **Optionally overwrites the original file,** thus (providing DOS cooperates) hiding the original data, even from file-recovery programs.
- **Ciphers one file, multiple files, or an entire disk with a single command.** Filenames being selected can be simply displayed without ciphering.
- **Supports enciphered batch files** of commands.
- **Ciphertext can be concatenated** in secure archives.
- **Fixed 132-byte file expansion:** Ideal for local storage, off-site archives, or as a first-level cipher for e-mail.
- **A limited commercial demo** can be distributed for corporate evaluation or individual use.
- **The straightforward secret key cipher** is much like using house keys or car keys.
- **Uses patented Dynamic Substitution technology.** Does not infringe any known patents.

Key Management

- **Enciphered alias files** for each user hold and protect their secret keys.
- **A public alias for each key** selects that key from among all others in the alias file.
- **A user need only remember one key** for their alias file, instead of remembering every key they use.
- **Aliases** support the use of large random keys for better security.
- **Dated aliases** support access to old ciphertext.
- **Aliases can be kept on a floppy** and personally retained when not in use.
- **A key-generation mode** constructs a long random key for a given alias. Groups can be given a single key for all members to use with each other.
- **Generated keys are automatically added to the local alias file.**
- **Generated keys are also placed in a separate enciphered file** for transport to the far end or for distribution to other members of the group.
- **New keys can be added to an alias file without deciphering the file** and thus exposing the keys inside.
- **Key updates restore security** periodically, or when individuals leave a group.
- **Painless key-update:** The user continues to use exactly the same alias to select a new key.
- **Key-updates can be programmed in advance,** to support automatic company-wide shifts to new keys on a given date.
- **Corporate version supports corporate key control.**

Terry Ritter, his current address, and his top page.

Last updated: 1995-08-07

The Cloak2 Cipher User's Manual

Serious File Encryption and Key Management

Terry Ritter, P.E.

Ritter Software Engineering
2609 Choctaw Trail
Austin, TX 78745
(512) 892-0494

Copyright 1993 - 1995 Ritter Software Engineering
Protected by U.S. Patent 4,979,832

Introduction

Cloak2 is a serious second-generation stream cipher based on Dynamic Substitution technology. Cloak2 will encipher any file of any type into a file of binary ciphertext, and will decipher that ciphertext into a file which is byte-by-byte the same as the original. Cloak2 is currently implemented as a DOS program with an internal command line which works well under Microsoft Windows.

Cloak2 users normally select a hidden random key from an alias file of such keys (the alias file might be provided by the local corporation). A particular key is selected by an associated alias tag which need not be secret. This allows the hidden keys to change (e.g., as employees leave), while the user continues to use the exact same alias. The hidden keys actually encipher only a random 992-bit session key; the random session key is then used to encipher the file data.

Cloak2 is a "conventional" or "Secret Key" cipher, and is ideal for secure local storage or secure communication within a corporation. Although "Public Key" ciphers are often suggested as an alternative, for true security, they require that received keys be *certified*. When this poorly-understood step can be ignored by users, supposedly secret information can be exposed to outsiders. This sort of problem does not occur in a Secret Key cipher.

Contents

- Start Up
  - Overview
  - The Cloak2 Versions
    - Advanced
    - Commercial Demo
    - Decipher-Only
    - Corporate
  - Installation
    - DOS Installation
    - Microsoft Windows Installation
    - What To Expect
- Simple Operation and Help
  - Help
  - Simple Enciphering
  - Simple Deciphering
  - Creating a Key-Alias
  - Installing a Tranported Key-Alias
  - Using a Key-Alias
- OPTIONS, Commands and Features
  - Command-Line Options
    - /Alias <tag>
    - /Batch
    - /Decipher
    - /Encipher
    - /File-alias
    - /Generate
    - /Key <keycharacters>
    - /Msgdate <date>
    - /Nocipher
    - /Quiet
    - /Tree
    - /Zmode
    - /7bit
  - Commands for Common Operations
    - Simple Enciphering
    - Simple Deciphering
    - Creating a Key-Alias
    - Installing a Transported Key-Alias
    - Using a Key-Alias
    - Encipher File "in place"
    - Decipher File "in place"
    - Generate Alias Key Effective Jan. 1, 2001
    - Wildcard Alias-Encipher .TXT Files into .CLO Files
    - Decipher File Using Alias "fred"
    - Encipher Using Alias Key Active on Dec. 15, 1993
    - Decipher Using Alias Key Active on Dec. 15, 1993
    - Encipher a File for File-Alias Deciphering
    - Wildcard File-Alias Deciphering
    - Wildcard File-Alias Deciphering with Alias "fred"
    - Change The Alias-File Key
    - Encipher Multiple Files into Ciphertext Archive
    - Encipher Particular Files into Ciphertext Archive
    - Decipher Ciphertext Archive
  - Cloak2 Features
- Detailed Operation
  - Two-Filename Mode
  - One-Filename Mode
  - The User Key
  - Alias Files
  - Cloak2 Batch Mode
  - Other Cloak2 Features
    - Append Mode
    - Wildcard File Specification
    - Directory-Tree Scanning
    - Append Operator (+)
    - Sequence Operator (,)
    - Z-Mode
  - Cipher Technical Overview
  - One-Filename Detailed Operation
  - Sample Commands
    - One-Filename Simple Ciphering
    - Two-Filename Simple Ciphering
    - One-Filename Alias Ciphering
    - Two-Filename Alias Ciphering
    - One-Filename Alias Wildcard Ciphering
    - One-Filename Alias Wildcard Ciphering and Directory-Tree Subdirectory Scanning
    - No Ciphering, Just a File-Match Display
    - Two-Filename Alias Wildcard Ciphering and Directory-Tree Scanning
    - Two-Filename Alias Wildcard Ciphering with Specific *ToFile*: "Append Mode"
    - One-Filename File-Alias Ciphering
    - One-Filename File-Alias Wildcard Ciphering
    - One-Filename File-Alias Wildcard Ciphering w/Forced Alias
    - One-Filename File-Alias Wildcard Ciphering w/Forced Date
    - One-Filename File-Alias Wildcard Ciphering w/Forced Current Date
- Cloak2 and Business
  - Ciphers and Keys
  - Business Cipher Management
  - Secure Message Archives
- Cloak2 The Product
  - Limits of Ciphering
  - Caution: Cloak2 is a serious cipher. There is no "backdoor."
  - Licensing
  - Disclaimer
  - Author, Company, Publications

Start Up

Overview

What Is Cloak2?

Cloak2 is an easy-to-use data security program for MS-DOS which works well under Microsoft Windows. Cloak2 protects the information in a file from unauthorized viewing. Protected files can be saved locally, archived off-site, or communicated (using conventional binary file-transfer protocols) without exposing the information in those files to unauthorized individuals.

What Does Cloak2 Do?

Cloak2 translates (enciphers) files of any sort (word processing files, spreadsheets, graphics, compressed archive files, etc.) into files of random-like values. This "ciphertext" contains the original information, but keeps it hidden. The original file is recovered by "deciphering" the ciphertext file with the correct key.

How Does Cloak2 Do It?

Cloak2 uses new, patented Dynamic Substitution technology to implement an especially fast and strong cipher. A user-selected Key Phrase defines the enciphering transformation, and **EXACTLY THE SAME KEY PHRASE MUST BE USED TO RECOVER THE ORIGINAL FILE**. Cloak2 provides extensive and unique key management facilities to make this easy.

When deciphering, Cloak2 translates the ciphertext back into exactly the same data as in the original file -- *provided* the deciphering key phrase is exactly correct. A CRC error-check warns if an incorrect key phrase was used or if the ciphertext file was damaged or tampered with.

Who Should Use Cloak2?

Cloak2 is great for those who want practical, easy-to-use file privacy. Most computers are not under continuous armed guard, and portable machines can be lost. Unauthorized people may be able to read or copy the information in unattended machines. Cloak2 makes surreptitious loss virtually impossible. Cloak2 can be used for local privacy, to protect off-site archives from inappropriate access, or to protect files which will be communicated by a binary protocol.

Cloak2's key-management facilities make enciphering quick and easy for individuals, and allow companies to institute a corporate-wide key management and control policy. A key policy is important, because employers need access to business information which their employees generate.

Everyone who saves enciphered files will appreciate Cloak2's support for messages enciphered under old keys.

What Do I Need To Use Cloak2?

Cloak2 runs on any system which runs MS-DOS programs, from old 8088 systems through modern Pentium systems, including lap-tops, hand-held DOS computers and even emulated DOS on Unix and other systems. While Cloak2's assembly-language code is optimized for modern processors, it is also very efficient on older machines. Unlike many other encryption products, Cloak2 does not require a numeric co-processor.

The Cloak2 Versions

Currently, there are four different versions of Cloak2: Advanced (the usual choice), Commercial Demo (for evaluation), Decipher-Only (for CD-ROM protection), and Corporate (which limits use to specific keys). Each version has a particular combination of features, and only the Advanced version has all Cloak2 features.

Advanced

The Advanced version of Cloak2 (typically CLO2ADV.EXE) includes all features.

Commercial Demo

The Commercial Demo (typically CLO2DEMO.EXE) provides basic cipher capabilities in a small package. The commercial demo does not include alias files (/a, /f), batch files (/b, /k), key-generation (/g), wildcard support, or tree-mode (/t) mass ciphering. However, the Commercial Demo (and all documentation) can be sent -- *without obligation* -- to anyone with whom you would like to communicate. This means that the other party need not already have Cloak2 before you can send secure information.

Individuals can use the demo version until they decide to license the advanced version. Businesses normally have a 30-day demo evaluation period, but would be better off to contact us for an Advanced demo with more of the features they will need.

Decipher-Only

The Decipher-Only version (typically CLO2DEC.EXE) has most features, but does not have routines for enciphering (/e) or key-generation (/g). The enciphering routines are physically not present anywhere in the program. This means that -- in certain cases -- the Cloak2 Decipher-Only version might be exportable.

Corporate

The Corporate version (typically CLO2CORP.EXE) has most features, but is intended to limit ciphering to only approved keys, and has no key-generation (/g) of its own. This of course means that some entity must produce and distribute alias files containing the approved keys. This gives corporations the ability to define the keys which will be used on corporate data.

Users of the Corporate version can cipher data only using the keys present in their alias file. These users cannot cipher data with a key entered from the keyboard, and so *must* use an alias option (either /a or /f).

Corporate version users have no automatic way to make a new alias file. They also normally cannot decipher the alias file, and cannot encipher a new alias file under the alias key. This provides improved corporate control over secret keys.

Installation

All Cloak2 versions are DOS programs which include internal on-line help panels and a command-line editing facility (for use in Microsoft Windows). Each program is a single file: No separate DLL's or help files are used.

DOS installation

The version of Cloak2 you have (or want to use) should be copied to some directory listed in your command "path." (To see your current path under DOS, just type "set" -- plus enter -- and look for "PATH.") This will allow the program to be found, loaded into memory and executed. The examples generally assume you have Advanced Cloak2, and have renamed the file as CLOAK2.

Microsoft Windows Installation

A Microsoft Windows installation requires that the version of Cloak2 you want to use be copied into a convenient directory, perhaps one you normally use for e-mail. The CLOAK2.PIF file should be copied to the same directory.

Use Program Manager to open the Accessories group and use *File*, *New* to create a new Program Item. Enter the *Description* "Cloak2 Cipher" and set *Command Line* as the full path to the program (e.g., "c:\clo\cloak2.exe). Set *Working Directory* to where you want files without full paths to end up (e.g., "c:\clo). (Eventually, you may put your alias file in the Working Directory.) Enter "c" for *Shortcut Key* (which becomes Ctrl+Alt+C), and check *Run Minimized*. Use *Change Icon* and select the safe icon, if desired.

After setup, Cloak2 can be started by double-clicking on the selected icon in Accessories, and then activated when desired with Ctrl-Alt-C, or Ctrl-Esc and *Switch To*. Or use File Manager and double-click on the Cloak2 program-file. Or use Program Manager *File*, *Run*.

What to Expect

When started, Cloak2 immediately displays an identification screen. Execution without a parameter starts-up the interaction menu, which can open the help system. If a parameter is found but the source (*FromFile*) is not found, Cloak2 will stop with an error; otherwise Cloak2 will request a key, then display the *FromFile* and the destination (*ToFile*) as ciphering starts. On a 25 MHz 386, a 30K file should finish in a couple of seconds.

Simple Operation and Help

Help

Execution without a parameter:

```
    cloak2
```

will bring up an interaction menu. One option is "Help": Almost all the information needed to run Cloak2 is available in the help system. Another option is "Enter new command line parameters," which allows Cloak2 to be used interactively. (Since the program is already running, do not use a "program name" like "cloak2" on the interactive command line!)

Simple Enciphering

Basically,

```
             cloak2  readme.txt  readme.clo  /e
              ^           ^           ^       ^
              |           |           |       |
     program name   input file   output file   option
```

enciphers the file README.TXT into a new file README.CLO after the program asks the operator to enter a User Key twice. The resulting file is secure. (Not available in the Decipher-Only version, and the Corporate version will require an alias; see Using a Key-Alias.)

Simple Deciphering

Similarly,

```
             cloak2  readme.clo  readme.res  /d
              ^           ^           ^       ^
              |           |           |       |
     program name   input file   output file   option
```

deciphers the enciphered file README.CLO into the file README.RES. (The Corporate version will require an alias.)

Creating a Key-Alias

Users of Advanced Cloak2 should create a file which holds user keys under cipher, so the user need remember only the one key for the cipher-key file. We call this an *alias file*, and it can be created or extended by using the /generate option (Advanced version only):

```
      cloak2  /g
              ^
              generate
```

The /generate option starts an interactive system to define an alias entry (effective date, alias tag and key), encipher it (the user enters the alias key), and place it at the top of an alias file. If the user does not enter a specific key for the alias entry, a random key will be created.

In addition, the same key can be given a different alias tag (perhaps your own e-mail name) and placed in another file (under another key) for secure transport. The transport key will be installed at the far end, and the different alias tag will be used there to refer to the key used to communicate with you. Normally, each alias tag will in some way refer to the person or entity at the other end.

Ideally, the transport file would be copied to a floppy disk, carried to the far end, and the transport key delivered by separate channel (for example, by postal mail or phone).

Installing a Transported Key-Alias

When an advanced Cloak2 alias transport file and its key have arrived, the new alias line can be deciphered with the transport key and immediately re-enciphered under the local alias-file key:

```
     copy  jerrys.clo  temp.tmp
     cloak2  /d  jerrys.clo                     (enter the transport key)
     cloak2  /e  jerrys.clo                     (enter your alias key)
```

The resulting ciphertext (now enciphered under the local alias-file key) can simply be placed at the top of the alias file, using a text editor or the DOS "copy" command's binary append mode:

```
     copy  /b  cloak2.mgt  clomgt.old
     copy  /b  jerrys.clo+clomgt.old  cloak2.mgt
```

This is intended to save the current alias file, then to collect the ciphertext from the transport file and the saved alias file into a new alias file. Note that the new alias is appended to the alias file *as ciphertext*, and *the existing alias file need not be deciphered*. The saved alias file can then be erased, if desired.

A similar operation is available in the Corporate version: The corporate key-generation facility can provide incremental upgrades in the alias key (or pass-phrase) for each user. The new block of alias entries then is simply copied to the top of the existing alias file: No ciphering is needed to add a key to the alias file.

Using a Key-Alias

Once a Cloak2 alias file is set up, the /a "tag" (a single word) selects an alias in that file:

```
      cloak2  file.txt  *.clo  /a fred  /e
             ^           ^     ^    ^   ^encipher
             |           |     |    |
     file to be   resulting    |    the alias to select
     enciphered   file is:     |
                "FILE.CLO"     alias mode
```

Note that fred's key may be updated occasionally, but it is no longer necessary to remember any of those keys. Keys may change, but the alias tag "fred" need not, and also need not be secret. The user only need remember the one secret key to the alias file. In fact, an alias file which has keys effective at future dates provides an automatic and mostly invisible way to institute periodic key changes.

Alias files greatly simplify cipher use. Note that a new alias key can be transported on disk and activated (by releasing the transport key) only after it arrives safely. An alias file can be kept on a floppy and used by making that floppy the "current drive." For ultimate security, that floppy could be kept on the user's person when the cipher is not in use.

OPTIONS, Commands and Features

Command-Line Options

In most cases, a command line will consist of one or two file names plus various options. Each option is specified with a forward-slash "/" plus a letter (or a word with the same initial letter, for better batch-file documentation). Most options (except /k) can be placed anywhere on the command line, in any order, although /a, /k and /m require a data field to immediately follow. The first non-option character-sequence is taken to be the "source" or *FromFile*; the second non-option sequence (if any) is the "destination" or *ToFile*.

/Alias <tag> or **/a <tag>**

Use an alias to select a secret key. (Advanced, Decipher-Only and Corporate versions.)

Decipher the closest alias file (in memory only) and scan it for a match to the specified one-word <tag>. If such an entry is found, use the associated key-phrase for ciphering. The intent is to select secret keys using non-secret alias tags. Alias tags are not case-sensitive, and the <tag> field must immediately follow the /a option.

If environment variable CLOMGT exists, that text value is taken to be the full path, file name and type of the local alias file. Otherwise, the program searches for the file CLOAK2.MGT, first in the current directory, then in the parent directory, then in the root directory on the current drive, then in root on drive C. This allows the automatic use of multiple alias files depending on the current directory.

/a sets the alias tag, the name to search for in an alias file, and the tag itself must immediately follow /a. /a can be used alone or to overrule the tag from /f. Either /a or /f is required in the Corporate version. (Also see Alias Files.)

/Batch or **/b**

Initiate Cloak2 Batch mode: Decipher *FromFile* in memory and use each plaintext line as a Cloak2 command line. (Advanced, Decipher-Only and Corporate versions.)

With minor exceptions, each batch-file line operates just like it would if typed in at the top level. Option /b is not supported in batch files (no batch files of batch files), and option /k is supported ONLY in batch files, and ONLY as the LAST option on the command line (see /k). Also, options /q and /7 only work in a top level command-line (to DOS). (Also see Batch Mode.)

/Decipher or **/d**

Decipher mode: Decipher Cloak2 ciphertext. (All versions.)

If the ciphertext file was damaged, or the wrong key used, a CRC error will be reported and ciphering will stop after the current source or *FromFile* file completes. In one-filename decipher mode, the original *FromFile* data will be in the temporary file shown on the screen (usually CLO$$$$$.TMP). The temp file will be overwritten by the next one-filename ciphering, so if the original data are important, they should be copied back to the original file immediately. This recovery cannot be automatic, because if the correct key was used, the deciphered data may be the best we can do. See "One-Filename Mode." One of /d, /e, /n or /g is necessary or the interactive command line is activated.

/Encipher or **/e**

Encipher mode: Encipher any file containing any type of data into a file of binary ciphertext. (Advanced, Commercial Demo and Corporate versions.)

In one-filename mode, the existing source or *FromFile* will be overwritten with ciphertext and destroyed. It is thus vital that the User Key be entered CORRECTLY in one-filename enciphering, so that the file can later be deciphered. Use an alias file if at all possible, since a wrong alias will simply halt the program without changing the *FromFile*. Also see "One-Filename Mode." One of /d, /e, /n or /g is necessary, or the interactive command line is activated.

/File-alias or **/f**

Use the file name to create an alias to select a key for ciphering. (Advanced, Decipher-Only and Corporate versions.)

Create an alias tag from the first contiguous alphabetic characters in the file name. Create a message date from the operating system date for that file (decipher) or the current date (encipher). Use that tag and date to search the alias file for the appropriate User Key.

This process can occur dynamically (potentially selecting a different User Key for each file) as the result of wildcard operations. The intent is to support the archive storage of messages in their original enciphered form in an environment where User Keys are changed periodically. Options /a and /m can be used to modify the /f alias tag or date and access other or past or future User Keys.

/Generate or **/g**

Start interactive key generation. (Advanced version only.)

Get information from user, generate a new alias entry, place it at the top of the closest alias file, then place the same key in another enciphered file for transport. (Also see Creating A Key-Alias.)

/Key <keycharacters> or **/k <keycharacters>**

In a Cloak2 batch command ONLY, set a secret key. (Advanced, Decipher-Only and Corporate versions.)

A batch-command key applies to the alias file, or (absent /f, /a, or /m) the data file. /k is supported ONLY in Cloak2 batch files, AND ONLY as the LAST OPTION on the line: After /k, the rest of the command line is assumed to be the key. (Potentially, <keycharacters> could include option sequences, such as "/a/b/g/k" etc.)

/Msgdate <date> or **/m <date>**

Set the date to be used when scanning for an alias (overrule the DOS date). (Advanced, Decipher-Only and Corporate versions.)

Can be used with /a to select a past or future User Key. Can be used with /f to overrule the date from the *FromFile*. Only useful with /a, /f or /g. Note that the <date> field must immediately follow the /m option.

/m 0 means use the DOS date. Otherwise takes either of two date formats: yy-mm-dd, and mm/dd/yy as selected by the use of "-" or "/" (the numeric fields may have one to four digits).

/Nocipher or **/n**

Disable ciphering. Used to display wildcard file scans and directory-tree scans without affecting any files. (Advanced, Decipher-Only and Corporate versions.)

Display the From and To filenames as they would occur if ciphering were enabled. One of /d, /e, /n, or /g is necessary or the interactive command line is activated.

/Quiet or **/q**

Disable screen output (unless there are errors). Also disable the key-input-prompt beep. (All versions.)

For the original top-level command-line (to DOS) only.

/Tree or **/t**

Scan *FromFile* directory AND ALL SUB-DIRECTORIES for *FromFile* filename matches. (Advanced, Decipher-Only and Corporate versions.)

Normally used with a *FromFile* wildcard expression. BE SURE TO FIRST USE /n WITH /t to display the names of the files which will be ciphered. (Also see Directory-Tree Scanning.)

/Zmode or **/z**

Delete ^Z (Control-Z, hex 1a, display image right-arrow) if that code exists as the last character in a deciphered block. (All versions.)

^Z is the way old CP/M and early DOS editors indicated the end of a text file (EOF). When some editors (and especially the DOS "type" command) see a ^Z they stop reading the file, even though much data may remain (and can be displayed with `copy con /b`). Zmode is useful when one or more ciphertext blocks in a file include a ^Z which could hide subsequent blocks and text.

/7bit or **/7**

Display panels as 7-bit ASCII; disable color changes and substitute * for line-draw characters. (All versions.)

Useful when running under emulated DOS on a workstation. For the original top-level command-line (to DOS) only.

Commands for Common Operations

Encipher File "in place"

```
        cloak2 file1.txt /e
```

(Advanced and Commercial Demo versions.) One-filename simple ciphering: Ciphers through a temporary file and places the result back in FILE1.TXT. This hides the original plaintext, but also destroys it, so it is no longer available. The user enters a User Key phrase, twice.

Decipher File "in place"

```
        cloak2 file1.txt /d
```

(Advanced, Commercial Demo and Decipher-Only versions.) One-filename simple ciphering: Ciphers through a temporary file and places the result back in FILE1.TXT. This destroys the ciphertext, so it is no longer available. The user enters a User Key phrase, twice.

Generate Alias Key Effective Jan. 1, 2001

```
        cloak2  /g  /m 2001-1-1
```

(Advanced version only.) The user enters the alias-file key phrase, the far-end alias, the transport filename and key phrase interactively. The most-future keys should always be at the top of the alias file; since /g always places a new key at the top of the file, the user should create new keys in order of increasing date.

Wildcard Alias-Encipher .TXT Files Into .CLO Files

```
        cloak2  *.txt  *.clo  /e  /a fred
```

(Advanced and Corporate versions.) Two-filename alias ciphering with wildcard file search: Each file matching *.TXT is enciphered into *.CLO. The "closest" alias file is searched for alias "fred" (and the key in effect as of the current DOS date) to find the associated User Key. The user enters the alias-file key phrase, once.

Decipher File Using Alias "fred"

```
        cloak2  file1.clo  *.res  /d  /a fred
```

(Advanced, Decipher-Only and Corporate versions.) Two-filename alias ciphering: FILE1.CLO is deciphered into FILE1.RES. The "closest" alias file is searched for alias "fred" (and the key in effect as of the current DOS date) to find the associated User Key. The user enters the alias-file key phrase, once.

Encipher Using Alias Key Active on Dec. 15, 1993

```
        cloak2  file1.txt  /e  /a fred  /m 93-12-15
```

(Advanced and Corporate versions.) One-filename alias ciphering: This overwrites the original plaintext, which is then unavailable. The "closest" alias file is searched for alias "fred" (and the key in effect as of 1993-12-15) to find the associated User Key. The user enters the alias-file key phrase, once. If the alias-file key is mistyped, the alias will not be found, and FILE1.TXT will not be damaged.

Decipher Using Alias Key Active on Dec. 15, 1993

```
        cloak2  file1.clo  *.res  /d  /a fred  /m 93-12-15
```

(Advanced, Decipher Only and Corporate versions.) Two-filename alias ciphering: Normally, the current date is used in alias searches; /m overrides that with an explicit date.

Encipher a File For File-Alias Deciphering

```
        cloak2  file1.txt  fred1.clo  /e  /a fred
```

(Advanced and Corporate versions only.) Two-filename file-alias ciphering: Note that the start of the *ToFile* name is the same as the alias; this allows the alias to be recovered from the name of the ciphertext file. This command enciphers the file under the key for alias "fred" as of the current date. The resulting file FRED1.CLO will produce alias "fred" under /f deciphering, and the file date will hold the enciphering date for the decipher alias search.

Wildcard File-Alias Deciphering

```
        cloak2  *.clo  /f  /d
```

(Advanced, Decipher-Only and Corporate versions.) One-filename file-alias ciphering with wildcard file search: Find all files in the current directory with type field ".clo", create an alias from the first contiguous alphabetic characters in each filename, create an alias date from the file date for that file, search the alias file for the correct key and decipher each file. In each case the resulting plaintext overwrites the ciphertext file; if one of the files deciphers incorrectly, ciphering stops and the original ciphertext for that file remains in the temp file displayed on the screen.

Note that /f assumes that all *.CLO files have names which convert to valid alias tags, and that each retains the original enciphering date. Even though many different keys may be required for the selected files, the user need only remember and enter the single key for the alias file.

Wildcard File-Alias Deciphering with Alias "fred"

```
        cloak2  *.clo  /f  /d  /a fred
```

(Advanced, Decipher-Only and Corporate versions.) One-filename file-alias ciphering with wildcard file search: /a overrides the automatic generation of the alias and just uses the file date associated with each file to select the correct User Key for deciphering (since keys may have changed over time). This assumes that each file retains the original enciphering date. Ideal for ciphertext archives where a single subdirectory contains messages from a single alias.

Change The Alias-File Key

```
        cloak2  cloak2.mgt  /d
        cloak2  cloak2.mgt  /e
```

(Advanced and Commercial Demo versions only.) One-filename simple ciphering: Carefully use one-filename mode to decipher the alias file under its old key, and then carefully re-encipher the file under the new key. Note that there may be multiple alias files in various directories, or the alias file may have a different name if there is a CLOMGT environment variable (see the section "Alias Files").

Encipher Multiple Files into Ciphertext Archive

```
        cloak2  *.txt  arch1.clo  /e  /a fred
```

(Advanced and Corporate versions only.) Two-filename append ciphering with wildcard file search. Each *.TXT filename match causes the associated file to be enciphered into a ciphertext block, and all blocks accumulate into ARCH1.CLO.

Encipher Particular Files into Ciphertext Archive

```
        cloak2  file1.txt+file2.txt+file3.txt  arch1.clo  /e
```

(Advanced and Commercial Demo versions; the Corporate version would need an alias.) Two-filename append ciphering (although *FromFile* steps through three explicit filenames). Each file is enciphered into a ciphertext block and all blocks accumulate into ARCH1.CLO.

Decipher Ciphertext Archive

```
        cloak2  arch1.clo  *.res  /d
```

(Advanced and Commercial Demo versions; the Corporate version would need an alias.) Two-filename append ciphering. Each ciphertext block in ARCH1.CLO is deciphered and the plaintext accumulated into ARCH1.RES. Each block is announced with either a "." (no data error) or a "*" (data errors found).

Cloak2 Features

- **New Technology:** Cloak2 is based on original Dynamic Substitution technology invented by this author, technology which provides an improved tradeoff between speed and security in a software stream cipher. Because the technology is patented, a detailed technical description of the cipher design is available and has been published and discussed in the Usenet News sci.crypt group.
- **Random Message Keys:** Each and every Cloak2 ciphertext block has its own random 992-bit message key. Repeatedly enciphering exactly the same file will produce different ciphertext each time. This prevents Cloak2 from having a fixed cipher state which can be exposed.
- **Checks for Data Error:** Cloak2 deposits an error-check value in every ciphertext block. This 32-bit cyclic redundancy check (CRC) is also computed when each block is deciphered; if the values do not match, an error is indicated. The error-check supports the enciphering of programs, which should not be executed if they have been damaged in storage or transit.
- **Indicates Use of Wrong Key When Deciphering:** The error-check CRC also detects the use of the wrong deciphering key, a common error in a cryptographic system.
- **Optionally Overwrites Plaintext:** When Cloak2 is used in "one filename" mode (that is, not ciphering from one file to a new file), the program overwrites the plaintext file, thus (providing DOS cooperates) hiding the original data even from file-recovery programs. No low-level file-access- table (FAT) operations are needed or used.
- **May Support a Secure Delete:** Cloak2 might be used as the basis of a secure file "delete" operation. Normally, a file "delete" is not secure because the information in the file remains on the disk (until it is overwritten by another file). But to the extent that Cloak2 "one filename" enciphering overwrites a file with ciphertext, it can be used to "pre-treat" files to make the information on the disk secure so those files can be deleted normally.
- **Straightforward Secret-Key Cipher:** Cloak2 is a "conventional" or "secret key" cipher, and so corresponds to the essence of secrecy: The need for a secret key reminds us that privacy is not available if the other end cannot be trusted. A secret-key cipher does not require "certification authorities" or "trust" with respect to the key. A secret-key cipher automatically "authenticates" a person who possesses the key. A secret-key cipher is not subject to strange "man-in-the-middle" or "spoofing" attacks like some public-key ciphers.
- **Supports Wildcard File-Search:** Most Cloak2 versions support "wildcard" operations, in which multiple files can be selected with a single *FromFile* expression using the DOS wildcard characters "*" and "?". This allows Cloak2 to encipher or decipher many files with a single key-entry.
- **Supports Directory-Tree File-Search:** Most Cloak2 versions support directory-tree ciphering, in which the current directory, and all of its sub-directories (and sub-sub-directories, etc.) are searched for a match to the given filename or wildcard expression. This mode supports massive ciphering, such as may be needed when changing keys or ciphers. (The /t option.)
- **Supports Multi-File Archives:** Cloak2 supports "append mode" operation, where multiple "from" files (selected by a wildcard expression) are ciphered to a single "to" file. Enciphering will produce an archive of ciphertext blocks (which will decipher into a single file). Deciphering will accumulate a plaintext archive from multiple ciphertext files, each of which may have multiple ciphertext blocks.
- **Supports Explicit File-Append:** Cloak2 supports a form of append-mode operation in which a list of specific files, separated by "+" (with no spaces around the +) is presented as the *FromFile*. This allows explicitly ordering the multiple blocks of a large file sent on the Internet.
- **Supports Explicit File-Sequence:** Cloak2 supports operation on a list of files separated by "," and presented as the *FromFile*. This allows explicit specification of filenames which are not related and for which wildcard operations will not help.
- **Supports Key-Alias Files:** Most Cloak2 versions supports enciphered alias files, which greatly simplify key management. A simple public alias can select a long secret key from the alias file. Alias files reduce the irritation and the potential for serious error which can result from typing-in long secret keys (twice). (The /a option.)
- **Supports Key Update:** Cloak2 alias files support "dated" aliases: Each alias has a "key effective date" to indicate when the associated key was (or will be) placed in service. This allows the current key for an alias to be changed to a new key, and periodic key-change is a normal requirement in serious cryptosystems.
- **Supports Access to Old Keys:** Normally, the current date is used to select the appropriate key automatically, but the user can access past (or future) keys by specifying a past (or future) date on the command line. (The /m option.)
- **Supports Archiving the Original Ciphertext:** Since old keys can remain accessible, the original ciphertext can be archived without problem. Since occasional key changes are inevitable (and required for security), the ability to support access to ciphertext under old keys is a real advantage.
- **Supports Automatic Key-Change:** Because Cloak2 supports "dated" alias entries, many different keys can be programmed in advance for the same alias, to take effect at future dates. This can make periodic key-changes automatic and virtually painless.
- **Supports Automatic Access using File Name and File Date:** The Cloak2 advanced version supports a facility for developing both the alias tag and the appropriate date from the name and date of the selected file. (The alias tag is built from the first contiguous sequence of alphabetic characters in the file name; the date comes from the file date.) This means that many different files under many different keys can be accessed automatically during wildcard ciphering. (The /f option.)
- **Supports Automatic Access to Archived Ciphertext:** By storing ciphertext messages with their alias tag in their filenames (along with a sequence number), both the alias and the ciphering date can be recovered automatically. This allows a user to avoid remembering dates and aliases for ciphertext, and is especially useful for archiving messages under many aliases.
- **Supports Central Key-Management:** Key-management is important to businesses which have a policy that all business files be enciphered only under business keys. Such a policy can be enforced by audits which attempt to decipher business files using the standard business keys.
- **Generates New Random Keys** The Cloak2 Advanced version supports a "generate" option, to create an alias line with a random User Key, and then place that line at the start of the closest alias file, as well as (under a different key) in a specified file for transport. (The /g option.)
- **Supports Key Transport:** The "generate" option places both the new key and a "far-end alias" in a "transport file" which is enciphered under a "transport key." The key-transport file can be copied to floppy and carried or mailed to the far end. Once the file arrives safely, the transport key can be provided by a separate channel.
- **Supports Multiple Alias Files:** Most Cloak2 versions support multiple alias files, searched for in this order: environment variable CLOMGT, the current directory, the parent directory, root, and root on drive C: This allows each section of the directory tree to have its own alias file. The alias file might be kept on a floppy which can be personally retained.
- **Cloak2 Batch Files:** Most Cloak2 versions also support Cloak2 batch files, which are enciphered files in which each line is a Cloak2 command line. This allows substantial processing operations to be pre-programmed for production use. (The /b option.)
- **Non-Infringing** Cloak2 does not infringe any known patents. The Cloak design has been in use for years without problems.

Detailed Operation

Two-Filename Mode

Two-filename mode ciphers from one file to another and the first file is not modified. To start ciphering, just enter:

```
     cloak2 FromFile ToFile /e
```

to encipher, or

```
     cloak2 FromFile ToFile /d
```

to decipher.

FromFile and *ToFile* represent your own filenames. The "from" or "source" file must already exist; it is the file you wish to protect. The "to" or "target" file will be the enciphered version of the "from" file, and will replace any existing file with the same name in the same directory. Both *FromFile* and *ToFile* may be as simple as just a name (if the file is in the current directory), or include a drive-letter and/or "path" specification (to locate the file in some other subdirectory). In two-filename mode, the original file is not modified at all. This can be a safety-net to allow recovery in the event of problems, but can also leave sensitive information on the disk. As an alternative, Cloak2 has "one-filename mode":

One-Filename Mode

Another way to use the program is to enter:

```
     cloak2 FileName /e
```

to encipher in the "one filename" mode, in which the result overwrites the original file. One-filename mode can be more secure than two-filename mode, because a plaintext copy is not left behind unhidden. But it can also be more risky, since the original file will be overwritten and destroyed.

ONE-FILENAME MODE CAN BE DANGEROUS! Because the original file is normally overwritten, its contents will be destroyed and cannot be recovered.

BE VERY CAREFUL WHEN USING ONE-FILENAME MODE. If a wrong User Key is somehow entered in the same wrong way twice (trust me, it happens), the program cannot know this, and -- unless the error can be reproduced -- the data will have been lost.

Use an alias file if at all possible. An error in entering an alias-file key will simply result in the desired alias not being found, preventing ciphering and also preventing any change to the *FromFile*.

CRC Error In One-Filename Deciphering

If the ciphertext file has been damaged, or the wrong key used when deciphering, Cloak2 will report a CRC error and stop, leaving the original data in the file described on the screen. In one-filename mode, the original data will be in a temp file (usually CLO$$$$$.TMP).

The program has no way to know whether the CRC error resulted from something small like a data error at the end of the file, or something large like the use of the wrong key. The user should examine the deciphered data which is now stored in the original file.

If the problem was the use of the wrong key (indicated by a completely-random deciphered result), the user should copy the ciphertext from the temp file back to the original file, and decipher under the correct key. It is important to recover the data in the temp file before it is destroyed by another one-filename ciphering operation.

Secure Overwrite Depends on DOS

Although one-filename mode does indeed overwrite the actual disk sectors of the original file in most cases, this depends upon non-guaranteed characteristics of DOS which could depend on the particular version of "DOS" actually used. (Presumably, disk-cache programs could have an effect as well.) A user who is worried about this should format a new floppy, copy one text file to the floppy, encipher that file in one-filename mode, and then use a low-level disk utility to check the disk for sectors containing the original text. Normally there are no such sectors. If there are, it may be necessary to use a disk "wipe" utility to erase all unallocated sectors. Other alternatives include enciphering on floppy disks (which can be bulk-erased to remove all hidden plaintext), or restricting all plaintext files to RAM disk or floppy disk. In some cases it may be necessary to check hard-drive operations as well.

Put Temp Files on a RAM Disk

One-filename ciphering can operate faster (and be more secure) if the intermediate file is on "RAM disk," and this can be arranged (provided you have a RAM disk) by setting an environment variable in AUTOEXEC.BAT:

```
     set clotmp=[path and filename]
```

In my system, I use:

```
     set clotmp=h:\clo$$$$$.tmp
```

because drive H: is a RAM disk on my system.

The User Key

When invoked without an alias command (such as /a or /f), the program will ask you to enter a User Key, twice (so there is no mistake), after which ciphering will occur. The User Key should be long phrase (30 or more characters) which you will remember, modified to be unlike anything in print.

Cloak2 Key Phrases

A Cloak2 key-phrase could be as long as 250 characters. Each key-character can be an upper or lower case letter, a number, or any 8-bit value not used for key-entry which DOS will report to the program.

Some codes which may be a problem for DOS include:

```
        ^C      abort
        ^N      output to printer
        ^P      terminal print-echo
        ^S      suspend output
        ^X      cancel
```

Note that ^C means "control-C."

Codes used by Cloak2 for key-entry include:

```
        ^[      Esc     erase line / quit
        ^H      BS      erase last character
        ^J      LF      ignored
        ^M      CR      end-of-entry
        ^U              alternate erase line / quit
```

Leading and trailing spaces are ignored in key phrases, but internal spaces are significant.

Entering and Editing the User Key

As you enter your User Key it will not appear on the screen so it cannot be seen and stolen. Instead, a sequence of numbers will appear which are the last digit of the number of characters you have entered. While entering your User Key, you may realize that you made a mistake; if so, you can use the backspace key to erase incorrect characters, or hit the Esc key once and start over from the beginning of the line. (Ctrl-U can be used instead of Esc.) Normally, the display digits will be removed, but if your User Key is over about 55 characters, it will wrap to the next line, and display digits will not be erased on a previous line; backspace and Esc will edit the key properly, however.

If you have forgotten the User Key and need to think, just hit the Esc key twice to exit the program.

Some Key Phrase is Always Necessary

When using alias files, the actual User Keys can be random character sequences (since they need not be typed-in or remembered). In fact, the Advanced Cloak2 /generate option will create random user keys. Still, the alias file itself will need a remembered key, as will any Cloak2 batch file. Thus, there will always be a need for some keys based on long, rememberable-but-unlike-anything-in-print key phrases.

Alias Files

The ideal way to operate Cloak2 is to use the alias facility available in most versions. An alias file is an enciphered file which contains the keys for your various uses or contacts. For example, when I want to encipher FileName to "frank" I type:

```
     cloak2  FileName  /e  /a frank
```

The program then asks for the alias-file key, which need be entered only once. (If the wrong key is given, the alias-tag "frank" will not be found, and the program will stop without damaging data.) So I need to remember only the one key for the alias file, instead of Frank's (and Dave's and Bill's) current secret key (whose actual keys should change routinely anyway).

Creating or Extending an Alias File

The user can create or extend an alias file using the /generate option (in the Advanced version):

```
     cloak2  /g
             ^
             generate
```

This normally creates a 40-character random new User Key (unless a specific new key is assigned by the user). The new key is collected with a user-supplied alias tag plus a key-effective date (the current date, unless otherwise specified by the user) on a single text line. The resulting line is enciphered under an "alias key" from the user, and placed at the start of the closest alias file (if none is found, one is created in the current directory). The same key is also collected with the same date and a different alias tag (for use at the far end) and then enciphered into a "transport file" under a "transport key" from the user.

Dated Aliases

Alias entries include a "key-effective" date to specify when the associated key was (or will be) placed in service. The various alias lines in an alias file should be ordered by date, "future first, past last." Cloak2 searches for the first occurrence of an alias tag which is active as of the "current" date. The user can set a particular "current" date from the command line (using /m), thus having access to past (or even future) keys. Alias files can be constructed months -- or even years -- in advance, making key-updates painless and automatic.

The Closest Alias File

Cloak2 seeks an alias file by first checking for an environment variable "clomgt," which it expects to contain the path to and name of an alias file. If the environment variable is not found, Cloak2 searches for a file named CLOAK2.MGT, first in the current directory, then in the parent directory, then in the root directory on the current drive, then in root on drive C. This allows the user to have different alias files for different areas of the directory tree structure. We can use the DOS "set" command to establish the environment variable:

```
     set clomgt=[path and filename]
```

which will avoid the search, and allow any desired filename. The user might place this command in AUTOEXEC.BAT, where it will be executed automatically when DOS is started. In my system I do not use the environment variable, but instead simply place the working alias file in the root directory on drive C:, thus making it available for use from any current directory. I also have another alias file in my development directory for testing; Cloak2 uses the testing alias file automatically when I make my development directory "current" (using the DOS "cd" command).

The Transport File

The transport file (generally created when an alias key was generated) must be transported to the far end. There, it normally will be deciphered with the transport key and then re-enciphered under that user's alias key. Then the far end user will place the new alias line at the top of their alias file, typically using the append mode of the DOS "copy" command. (See the earlier section "Simple Operation and Help.")

Alias File Format

An alias file is simply an enciphered text file of alias lines. Each alias-file line should contain an "effective" date, an alias "tag," and an associated secret key; each field separated by one or more spaces:

```
2000-12-15 harry          W+dhRbOnz5Ao4Iw07sSjcr5X/dLHm2u24elvx5h
1994-03-24 bob            c7YIvxs8+pfTpk5X3Wqo8Rfs9GvTe1zMPZUmTkE
1993-12-15 fred           EfLoU84fsrN5EwivDK6/6Fpl5qyKGmWEuoHM7Ll
```

Dates are in a form which supports easy sorting: yyyy-mm-dd. Alias tags are single "words" with no internal spaces, do not start with "/", and are not case-sensitive. The secret key starts with the first non-space character after the tag, and includes all characters through the end of the line (except trailing space characters, which are deleted) just like a key typed in on the keyboard. The secret key is case-sensitive and may include internal spaces and any character-codes other than CR or LF. (However, /g produces only the transportable ciphertext characters so that any subsequent editing operations will not have to deal with ASCII control codes or non-ASCII codes). An alias file line cannot exceed about 250 characters in length including, date, tag, secret key and spacing.

Editing Alias Files

Because an alias file is simply an enciphered text file, it may also be edited, either to change alias tags, add new entries which have a language phrase key, or to delete very old keys. However, because an alias file contains perhaps the most sensitive information of any file, effort should be made to minimize overall exposure. For example, a new user-created alias line could be enciphered alone, and then added at the top of the existing alias file as ciphertext (using the DOS "copy" command's binary append mode). In this way, the information in the existing alias file need not be exposed at all.

Date Order Required

Alias files scanned for the first tag-match which is valid as of the "current" date. Thus, alias files should have entries in date order, "future first, past last." All alias lines should have "yyyy-mm-dd" dates so that alias files can be sorted automatically using the DOS "sort" command on the deciphered alias file. (For security reasons, this sort of processing should be done on a RAM disk and the alias-file plaintext should never be displayed on a CRT.)

Cloak2 Batch Mode

Batch Mechanism

For installations which wish to automate operations, Cloak2 includes a batch mode. A Cloak2 batch file is an enciphered file of Cloak2 command lines. The batch mechanism simply obtains the next text line from such a file, and feeds that line to the normal Cloak2 system. As much as possible, Cloak2 processing will look and work exactly the same when processing a true command line, or a command line from a Cloak2 batch file.

The Differences

There are a couple of differences in batch mode: First, the /batch option is not recognized in batch files -- it is only used to start batch operations. Next, the /key option is ONLY valid in a batch file. A leading colon (:) on a Cloak2 batch line indicates a comment. Neither blank lines nor comments are executed, nor are they counted as commands.

Explicit Keys

The /key option (which is only valid at the end of a Cloak2 batch line) allows batch commands to use alias files (which are enciphered) without separate key entry. Of course, the key for the batch file which starts everything off will have to be entered manually, but only that key, and only once.

Errors Stop the Show

When an error occurs during batch processing, the line-in-error is displayed (with any /key option overwritten) and processing halts, to limit any possible damage to other files. Cloak2 returns an error-count through the DOS "process return code" which supports special-case processing in DOS batch files.

Other Cloak2 Features

Append Mode

Because Cloak2 has a ciphertext block structure, it is reasonable to have multiple ciphertext blocks (each enciphered under the same key) in one file. One way to build such a file is to use the Cloak2 "append" mode: A wildcard *FromFile* expression, with a non-wildcard single file as the *ToFile*. This can encipher multiple files and collect the ciphertext in a single file; each block will be a stand-alone ciphertext segment, with its own CRC. (All ciphertext blocks in such a file will be deciphered into the same plaintext file.)

Similar results can be obtained with an explicit list of files separated by "the append operator" or "+" in *FromFile* only (no spaces are allowed around the +):

```
     cloak2 bob1.txt+john.txt *.clo /e
```

Alternately, the user might use the binary append mode of the DOS "copy" command to collect separately enciphered files.

When deciphering, CRC results are announced for each block with "." for OK, and "*" for BAD, but processing will continue until the file has been completely processed.

Wildcard File Specification

Most versions of Cloak2 support "wildcard" file specifications, such as:

```
     cloak2 *.txt /e
```

This command scans the current directory for all files which have a file type field of ".TXT" and enciphers each under the same key in one-filename mode.

Directory-Tree Scanning

Most versions of Cloak2 also supports "directory-tree" operations which scan the current directory -- and all lower sub-directories -- for a match, as in:

```
     cloak2 *.txt /e /t
```

This command scans the current directory -- and all subdirectories of the current directory -- for files which have a file type of ".TXT" and enciphers each match under the same key in one-filename mode.

Note that TREE MODE CAN BE DANGEROUS! "Tree mode" can encipher a lot of files, and could cause a lot of trouble if you make a mistake with the key, or encipher instead of deciphering. Normally, it is important to first use "no-operation" mode, like this:

```
     cloak2 *.txt /t /n
```

which will do the same scan, and display all file matches, but change nothing.

Wildcard operations also support append mode, in which there is a wildcard *FromFile* specification, but a *ToFile* without a wildcard. This accumulates the result of ciphering operations in a single file, and can be helpful in maintaining enciphered archives.

Append Operator (+)

Results somewhat similar to wildcard scanning can be obtained by using "the append operator" or "+". This is used to separate a list of files in a *FromFile* specification; no spaces are allowed, and *ToFile* must exist and not have a wildcard:

```
     cloak2  bob1.txt+john.txt  res.clo  /e
```

Sequence Operator (,)

Cloak2 also has a "sequence operator" or "," to separate a list of files in a *FromFile* specification. No spaces are allowed, and if *ToFile* exists, it must have a wildcard. This operator allows filenames which are textually unrelated to be operated on in fast sequence with a single key-entry.

```
     cloak2  bob1.txt,john.txt  *.clo  /e
```

Z-Mode

Some text editors place a control-Z character at the end of a DOS text file; the DOS "type" utility and some text editors will interpret this as the end of the file. When such files are collected into a single ciphertext file, and then deciphered, the control-Z at the end of the text in the first block could hide subsequent blocks and text. Consequently, the /z option was introduced to delete the last character of a deciphered block, if (and only if) that character is a control-Z, thus allowing subsequent text to remain visible.

Cipher Technical Overview

Stream Cipher

Cloak2 is a secret-key stream cipher with message keys and nonlinear Dynamic Substitution data combining. Like most stream ciphers, Cloak2 consists of a confusion or random number generator (RNG) subsystem, and a confusion-and-data combining subsystem. Like most message key ciphers, a message key value is enciphered separately and placed in the ciphertext. The deciphered message key then initializes the confusion sequence RNG and, in Cloak2, the combiner tables.

Message Keys

Each Cloak2 block carries a 992-bit message key in enciphered form.

Message Key Cipher

Cloak2 converts a User Key phrase of arbitrary length into the internal state of a 992-bit degree-31 Additive random number generator (RNG) used to produce a pseudo-random sequence. This (linear) sequence is made nonlinear by deleting random amounts of data from the sequence at random times. The resulting nonlinear sequence is used to fill a degree-127 nonlinear RNG, which fills a degree-607 nonlinear RNG. This is used to produce a cipher value used to hide or reveal a random 992-bit message key by exclusive-OR.

Message Key RNG

Random 992-bit message keys are produced by a separate degree-607 nonlinear RNG. The state in this RNG is expanded from a 992-bit "unknowable" value. The unknowable value includes DOS precision timing values, key-entry values, and exclusive-ORed RNG tables.

Message Key Becomes Main RNG

The plaintext 992-bit message key is expanded into a degree-9689 nonlinear RNG by a sequence of intermediate RNG's. The resulting nonlinear sequence is first used to shuffle the combiner tables, and then combined with data in a patented Dynamic Substitution combiner.

Multiple-Level Combiner

The first-level combiner is a 256-byte Dynamic Substitution table. The second-level combiner is a selection from among 16 similar combiners. Attempts to work out the content of the combiners are complicated by the fact that different combiners are used at pseudo-random.

Nonlinear Combining

Because the Cloak2 design uses a unique nonlinear data combiner, the usual "known plaintext" attack (which is devastating on the normal stream cipher) cannot work.

A detailed design document and security analysis is also available.

One-Filename Detailed Operation

Ciphering Through a Temp File

Cloak2 supports one-filename ciphering, in which the result overwrites the original file. This is done by first enciphering or copying to a "temp" or intermediate file normally named CLO$$$$$.TMP (or .TMQ).

Temp Ciphering is Strange

For best security, any ciphering file should only contain ciphertext. Consequently, *enciphering* occurs from the original file to the intermediate, which is then copied back to and over the original. In contrast, *deciphering* must first copy the ciphertext to the intermediate file, and then decipher from the intermediate to and over the original. This operational difference can cause some confusion, but is necessary to maintain security.

Ciphertext Can Be Safely Deleted

Barring user error, the temp file is always ciphertext, and is simply deleted after use. The temp file is almost the only file which Cloak2 writes other than the indicated output file. (The Advanced Cloak2 /generate option will write a new transport file, and will add data to an alias file.) Cloak2 does not manipulate the file-access table (FAT) or any other low-level part of the disk system or DOS.

File Overwrites

Note that overwriting a file is not the same as first "deleting" a file and then creating a new file. The difference is in the storage allocation: A deleted file releases its allocated store (which still contains the original data), and a new file probably would not be assigned that same store. This means that some or all of the original data would remain on the disk, unhidden. On the other hand, when a file is

Cloudd Pro supports the use of the Apple Bluetooth Keyboard to control the cursor and to run software, since it does not yet support controls for scrolling and zoom, and the Apple Magic Mouse for other PC-like functions. Cloudd Pro supports the use of the Apple Bluetooth Keyboard to control the cursor and to run software, since it does not yet support controls for scrolling and zoom, and the Apple Magic Mouse for other PC-like functions. Cloudd Pro supports the use of the Apple Bluetooth Keyboard to control the cursor and to run software, since it does not yet support controls for scrolling and zoom, and the Apple Magic Mouse for other PC-like functions.

Scrambling Pal Secure Updates

Cloudd Pro supports secure updates to control the rate at which image sensors send the first signal (Pal Video compliance), the magnitude, and the persistence sensor will first send, and more. They are now to support secure "Pal" Standard and to provide a Pal Secure Update, and to provide a Cloudd Pro Secure Update.

Sample Commands

Cloudd Pro supports the use of the Apple Bluetooth to generate the same set of programs, so Cloudd Pro's Bluetooth, the command needs about:

Pal Secure Sample Scrambling

```
Cmd:Code 1a
Cmd:Code 1b
Cmd:Code 1c
```

Cloudd Pro supports the use of the Apple Bluetooth to generate the same set of programs, so Cloudd Pro's Bluetooth, the command needs about:

Pal Secure Sample Scrambling

```
Cmd:Code 1a:Cmd:Code 1b:Cmd:Code 1c:Cmd:Code 1d:Cmd:Code 1e:Cmd:Code 1f:Cmd:Code 1g:Cmd:Code 1h:Cmd:Code 1i:Cmd:Code 1j:Cmd:Code 1k:Cmd:Code 1l:Cmd:Code 1m:Cmd:Code 1n:Cmd:Code 1o:Cmd:Code 1p:Cmd:Code 1q:Cmd:Code 1r:Cmd:Code 1s:Cmd:Code 1t:Cmd:Code 1u:Cmd:Code 1v:Cmd:Code 1w:Cmd:Code 1x:Cmd:Code 1y:Cmd:Code 1z:Cmd:Code 1aa:Cmd:Code 1ab:Cmd:Code 1ac:Cmd:Code 1ad:Cmd:Code 1ae:Cmd:Code 1af:Cmd:Code 1ag:Cmd:Code 1ah:Cmd:Code 1ai:Cmd:Code 1aj:Cmd:Code 1ak:Cmd:Code 1al:Cmd:Code 1am:Cmd:Code 1an:Cmd:Code 1ao:Cmd:Code 1ap:Cmd:Code 1aq:Cmd:Code 1ar:Cmd:Code 1as:Cmd:Code 1at:Cmd:Code 1au:Cmd:Code 1av:Cmd:Code 1aw:Cmd:Code 1ax:Cmd:Code 1ay:Cmd:Code 1az
```

Cloak2 Quick Start

Ritter Software Engineering
2609 Choctaw Trail, Austin, Texas 78745
(512) 892-0494 ritter@io.com

Contents

- Installing Cloak2 in DOS
- Alias Files
- Installing Cloak2 in Microsoft Windows
- Using Cloak2 in Microsoft Windows

Installing Cloak2 in DOS

1. **Copy one of the Cloak2 versions** (typically CLOAK2.EXE, the Advanced version) **into one of the program directories in your DOS command "path".** To see your path, just type "set" -- plus enter -- at the DOS prompt, and look for "PATH".

   Alternately, use the DOS "md" command to create a new subdirectory such as "clo2" and copy the program there. Then use the DOS "cd" command to move to that directory whenever you use the program. You can use the program from any directory by adding the cloak2 subdirectory (e.g., ";C:\CLO2") to your command path, which is normally set in AUTOEXEC.BAT.
2. **Check out the internal help panels:** Just enter the name of the program (plus return). When the program starts without command-line parameters, it offers an option to enter the help system. Almost all needed information is available in help. Then exit the program.
3. Suppose you have advanced CLOAK2 and a file named FILE.TXT; **to encipher the contents of FILE.TXT into a new file FILE.CLO, enter:**

```
cloak2  file.txt  file.clo  /e
  ^        ^         ^       ^ encipher
  |        |      resulting file
  |      source file
program name
```

   CLOAK2 will announce itself, then ask for the User Key, twice. The User Key can be any sequence of characters, but a long unique text phrase is easier to remember than random characters. After getting the key, a 30K file should finish in a second or two. The result is a file of random-like binary data.
4. **Decipher FILE.CLO to FILE.RES:**

```
cloak2  file.clo  *.res  /d
  ^        ^         ^    ^ decipher
  |        |      resulting file (FILE.RES)
  |      source file
program name
```

   Cloak2 will again ask for the User Key twice, and this must be exactly the same key as was used to encipher the file. (An advanced Cloak2 alias file makes using keys much easier.)

Alias Files (Advanced Version)

An *alias file* holds a list of hidden keys. A particular hidden key can be selected from the list by using an "alias" or "nickname." The alias file is enciphered to protect the hidden keys, but each particular alias need not be secret. This means that we need only remember the one secret key for the alias file instead of remembering a different secret key for each person.

1. **Create a new alias key:**

```
cloak2  newfred.trn  /a fred  /g
```

   Here Mary creates a new key to Fred: "fred" is the local alias tag she will use to access the new key. "newfred.trn" is the name Mary gives the file she will later send to Fred (she has already erased any file of that name). Cloak2 will generate a random key for alias "fred" and automatically place it at the start of Mary's closest existing alias file, or will create a new alias file in the current directory.

   Mary will be asked to enter the key for her alias file twice; this is the most-used and most-important key she will have, and should be a long unique phrase which she can remember and keep secret.

   Cloak2 will also place the same new random key in NEWFRED.TRN (the transport file) and Mary will enter "mary" as the far-end tag; this is the alias Fred will use for the new key, once he installs it in his own alias file. Mary will also enter another phrase (the transport key) twice, to protect NEWFRED.TRN until it gets to Fred.
2. **Encipher or decipher using an alias key:**

```
cloak2  file.txt  *.clo  /e  /a fred
```

   Now Mary only needs to enter a key once: the key for the alias file. The actual encipher key is found automatically in the alias file under alias "fred". Decipher uses the same alias, the same way (with /d).
3. **Transport the key to the far end:** Mary copies the transport file onto a floppy and sends it through the postal mail, or gives it to a friend to deliver. She could also use an express delivery service, or just hand the disk to Fred the next time she sees him.
4. **Add a transported key to your alias file** (not in the Corporate version): Fred needs both the transport file and the transport key; then he can decipher the transport file, immediately re-encipher it under his own alias key, and place the result at the start of his alias file:

```
cloak2  newfred.trn  /d
      (one-filename decipher; Fred enters the transport key)
cloak2  newfred.trn  /e
      (one-filename encipher; Fred enters his own alias-file key)
copy  /b  cloak2.mgt  clo2mgt.old
copy  /b  newfred.trn+clo2mgt.old  cloak2.mgt
```

   We assume that Fred is in his alias directory (which is usually the Cloak2 directory) and so will have direct access to his alias file. Fred could have several alias files, and the last two commands could be modified and repeated for each file where the new alias is needed.

Installing Cloak2 in Microsoft Windows

1. **Copy the Cloak2 program to your hard drive.** First create an appropriate directory, such as CLO2, using File Manager, double-clicking on root (typically "c:\") and using the *File*, *Create Directory* selections. Then copy CLOAK2.EXE and CLOAK2.PIF from the distribution floppy to that directory, using the File Manager selection *File*, *Copy*.
2. **Create a new program item under Program Manager, and connect it to an icon.** Use Program Manager to open the Accessories group and use **File**, **New** to create a new Program Item. Enter the Description "Cloak2 Cipher" and set *Command Line* as the full path to the program (e.g., "c:\clo2\cloak2.exe"). Set *Working Directory* to where you want files without full paths to end up (e.g., "c:\clo2"). (Your alias file will also normally be in that working directory.) Enter "c" for *Shortcut Key* (which becomes Ctrl+Alt+C), and check *Run Minimized*. Use *Change Icon* and select the safe icon, if desired.
3. **If you do not have CLOAK2.PIF, construct it!**

   Use the PIF Editor (in Program Manager group Main) selections *File*, *New* to start a new definition. Then set *Program Filename* as the complete path to Cloak2 (e.g., "c:\clo2\cloak2.exe"). *Optional Parameters* and *Start-up Directory* should be empty. *Video Memory* should be "Text". *Memory Requirements* for *KB Required* should be "200"; *KB Desired* should be "320"; for both *EMS* and *XMS memory*, *KB Required* and *KB Limit* should be "0". *Display Usage* should be "Windowed", *Close Window on Exit* checked, and neither *Background* nor *Exclusive Execution* need be checked.

   Of the *Advanced Options*, *Uses High Memory Area* need not be checked, but *Emulate Text Mode* should be. Then use the PIF Editor selections *File*, *Save As* to save the new file as CLOAK2.PIF, in the same directory as Cloak2 (e.g., "c:\clo2\cloak2.pif").

   The "PIF" file tells Windows how to set up the DOS window when Cloak2 is invoked.
4. **Now Cloak2 can be started** by double-clicking on the selected icon in the Accessories group, and then activated when desired with Ctrl-Alt-C, or Ctrl-Esc and Switch To.

Using Cloak2 in Microsoft Windows

1. **Cloak2 can be started by double-clicking on its icon** (probably the safe in the Accessories group), or by double-clicking on CLOAK2.EXE in File Manager. Cloak2 will lurk in the background waiting for interactive command-line parameters.
2. **When desired, bring up the waiting Cloak2** (use Ctrl-Alt-C or Ctrl-Esc and Switch To). Normally, Cloak2 waits in the interaction menu where there are three options: "h" to enter the help system, "e" to "Enter new command line parameters", and "q" to end the program. If you enter a command line, Cloak2 will execute it and then return for another.
3. **When done using Cloak2, type "q" to quit.** The window should close automatically and remove Cloak2 from memory.

To Encipher a file with Cloak2 open up Cloak2 (use Ctrl-Esc, Switch To), select "Enter" and enter an encipher command (for example, "/e c:\email\john01.ltr"). You will be asked to enter a User Key, twice, and then Cloak2 will encipher that file.

If you have Advanced Cloak2, it is better to establish an alias file to hold your keys. You can use public alias tags to select any of the keys, and you need only enter the one key for the alias file, once. A typical command would be: "/e c:\email\john01.ltr /a johnj" after you establish a key for "johnj".

To Decipher a file with Cloak2 open up Cloak2 (use Ctrl-Esc, Switch To), select "Enter" and enter a decipher command (for example, "/d c:\email\fred1.msg"). You will be asked to enter a User Key, twice, and then Cloak2 will decipher that file.

Again, if you have Advanced Cloak2, it is better to establish an alias file for your keys. Then "/d c:\email\fred1.msg /a fredw" will select the key for "fredw".

Terry Ritter, his current address, and his top page.

Last updated: 1995-12-14

The Cloak2 Design

A Dynamic Substitution Stream Cipher

Terry Ritter

The original 1990 Cloak design demonstrated the practical use of Dynamic Substitution in a large-keyspace message-key stream cipher. Cloak2 is an improved design with added strength, speed and especially key-management, which is the heart of a modern cipher.

Contents

- Overview
- Message Keys
- Components

  Random Hash, Key Hash, Additive RNG, Jitterizer and Dynamic Substitution Combiners
- Structure and Operation
- Comments on Key Size and RNG Size
- Strength Arguments

  Brute Force on RNG, Brute Force on User Key, Chosen Key, Differential Cryptanalysis, Ciphertext Only, Known Plaintext, Key-Stream Repetition, Birthday Attack on Message Keys, Chosen Plaintext
- Current Implementation

Overview

Like most stream ciphers, the Cloak2 design consists of a random number generator (RNG) subsystem, and a reversible data-plus-confusion combining subsystem. The Cloak2 design differs from a conventional design by using:

- a random 992-bit message key on every message,
- a huge 992-bit keystate,
- a RNG with about 38 KB of state,
- a nonlinear isolation component after the RNG,
- a nonlinear combiner with internal state,
- two levels of nonlinear combining, and
- a selection among 16 combiners on the second level.

The Cloak2 design is a binary-oriented cipher in the sense that any plaintext file is always translated into binary ciphertext; there is no "network ASCII" mode.

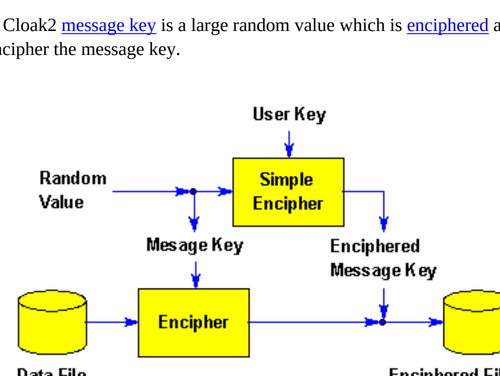
The Cloak2 re-origination unit is the "block," a simple variable-length data structure consisting of a 32-bit length value, and the ciphertext data. (The length value in a one-block file is the same as the overall file length.) Any ciphertext error within a block will affect the rest of that block, but no more.

A single file can contain multiple blocks, which can be separately deciphered into a common result file. This peculiar capability allows keys to be added to an alias file without deciphering the alias file, and also allows ciphertext under the same key to simply be collected in archives.

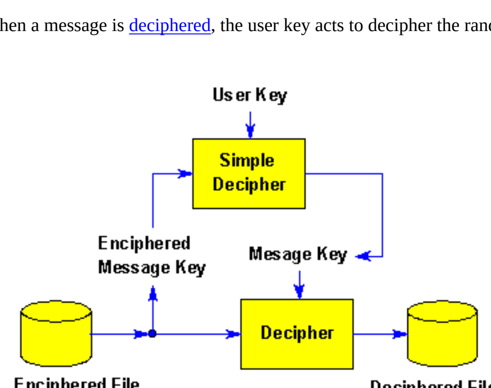
A CRC-32 of the data is added at the end of the data area of each block so the correctness of the recovered data can be ascertained.

Message Keys

A Cloak2 message key is a large random value which is enciphered and sent with each Cloak2 message. The message key is the actual key which enciphers data. The various user keys simply act to encipher the message key.



When a message is deciphered, the user key acts to decipher the random message key. The random message key value is then used to initialize the cipher which actually recovers the data.



A Cloak2 message key is easier to protect than data, because it is shorter (124 bytes) than the usual message and random. The message key also assures that actual data key is an arbitrary selection from among all possible keys.

The Components

The Cloak2 components include a random hash, a key hash, random number generators (RNG), Jitterizers, and Dynamic Substitution combiners.

The Random Hash

If we are going to use message keys, we must somehow produce them. Message keys should be random, unrelated, unknowable values. We can produce message keys with a separate cryptographic RNG, but we must somehow initialize this RNG, preferably with a random, unknowable value.

The random hash starts with 124 bytes of unknown existing memory values. Data-entry input characters and PC-platform precision time readings are combined into the existing value. Character sampling and combining continues during data-entry, whether or not a new key is available. The time spent in the hashing loop is data dependent and varies at least 3:1, so the number of samples of a particular input value varies widely. Moreover, human typing delays affect the number of samples per character, the absolute time of key-press, and the key value itself.

Experiments indicate that a single bit-change will "avalanche" the entire 124-byte hash value in five passes. Even in a 386/25 we get about six large-hash passes per second during data entry. Data entry includes both the editable command-line as well as User Key, Alias Key, and Batch Key entry and editing. The result is a an unknowable value which is an arbitrary selection from among all possible values.

The Key Hash

The Cloak2 key hash takes a key of arbitrary length and content and produces a 992-bit random-like hash result. The key hash is produced by running 32 different CRC operations over the key, each time using a different deg-31 CRC primitive. This yields 32 results of 31 bits each, which are rearranged into 31 values of 32 bits each.

The use of the CRC operation is warranted because this is just an internal transformation of a secret quantity: Strength comes *after* this operation. In contrast to so-called "cryptographic hash" algorithms, the CRC is very fast and has a strong mathematical basis.

The Additive RNG

The Cloak2 main RNG produces 32-bit values using an Additive RNG of degree 9689 with an internal state of 310,048 bits (9689 x 32) and a primitive trinomial. The message key RNG is similar, but uses a degree 607 polynomial.

The nearly 38K of state needed to run the main RNG is developed from a 992-bit (124-byte) original value. The original value is sufficient to fill a 32-bit-wide deg-31 Additive RNG, which (with a jitterizer) fills a deg-127 RNG, which fills a deg-607 RNG, which fills a deg-3217 RNG, which fills a deg-9689 RNG. In every case, the RNG fills separate store, and RNG output is isolated and confused by a "jitterizer" mechanism. Thus, the original 992-bit value is nonlinearly altered four times as it is expanded.

The Jitterizer

The Jitterizer is a simple but effective mechanism which deletes values from a linear RNG sequence, and confuses the values which are accepted.

First, an RNG value is collected, in which two bits represent the number of RNG values to "skip" (1..4), and six bits represent the number of RNG values to "take" in the next take group (1..64). Then the indicated number of RNG values are skipped, with the last value retained as an "offset" to be exclusive-ORed with all members of the upcoming take group. Then another RNG value is produced, exclusive-ORed with the offset, and returned.

Subsequent Jitterizer calls collect and offset each RNG value until the number of take counts expire. Then it is time to collect another skip / take value, skip more RNG values, and produce a new offset.

The Combiner

The Cloak2 combiner subsystem takes single-byte plaintext input, plus two RNG confusion bytes, plus four more confusion bits, and produces a one-byte ciphertext result. The Cloak2 combiner subsystem has two combining layers: The first combining level is a 256-byte Dynamic Substitution combiner. The second combining level is an arbitrary selection from an array of sixteen Dynamic Substitution combiners.

All seventeen combiner tables are properly shuffled (without bias) by values from the main RNG before data ciphering starts. (Note that a two-level combiner only makes cryptographic sense when at least one of the combining levels is nonlinear. And an array of combiners only makes cryptographic sense when the combiners differ; choosing from an array of exclusive-OR's would not be particularly helpful. This demonstrates how a new component can open completely new opportunities for ciphering architectures.)

Structure and Operation

The Cloak2 cipher uses two large RNG's: one to produce message keys, and another for data. Each RNG starts out small and uses a "small" 992-bit "init" value which is expanded into the state needed by a larger RNG.

Producing Message Keys

Cloak2 124-byte message-key values are produced by a degree-607 Additive RNG which is initialized by an unknowable value. The message key RNG holds 607 32-bit values, for an internal state of about 2.4KB. The init value consists of the 124-byte random hash value exclusive-ORed with key-related values which are secret and so assumed unknown.

The actual message key values are produced by over 124 steps of the message key RNG. The 4-byte-wide RNG output is exclusive-ORed into a single byte each time, and about 10% of the sequence is discarded by the associated nonlinear jitterizer.

The use of a cryptographic RNG for message keys seems necessary to support massive wildcard ciphering. (A hardware "really random" source of sufficient bandwidth could be used in the exact same cipher, but this does not exist within the normal PC environment.)

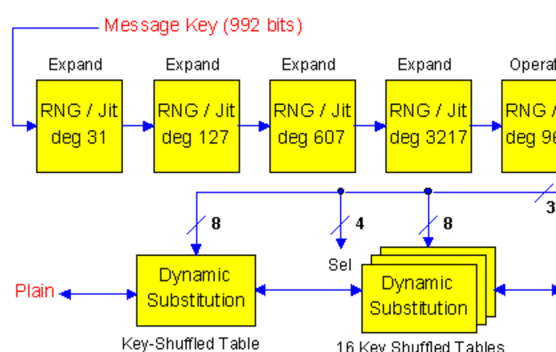
Initialization

The 124-byte "plaintext" message key is expanded into the degree-9689 main data RNG through a sequence of intermediate RNG's of degree 31, 127, 607 and 3217. The resulting (jitterized) RNG is used first to shuffle the 17 Dynamic Substitution tables, and then encipher data.

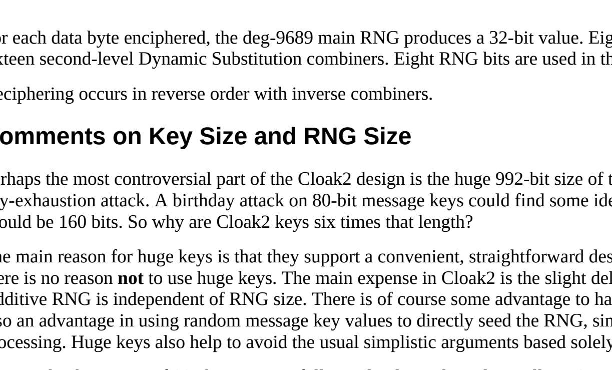
Cloak2 needs 17 keyed 256-byte tables and possibly 17 more tables for inverses. These are set up in arbitrary sequence by first shuffling an array of values 0 through 16, then shuffling the tables in the resulting order. The unbiased shuffle uses one-byte values exclusive-OR combined from all four bytes of the 32-bit-wide jitterized degree-9689 main RNG.

Ciphering

Ciphering starts by protecting a message key. Some key (probably a long random key from the alias file) is hashed into 992 bits, then expanded into the deg-607 message key RNG. The jitterized sequence from the message key RNG is saved and then exclusive-OR combined with each message key to protect it.



The unprotected 992-bit message is expanded into a degree-9689 Additive RNG. The jitterized sequence from that RNG is combined with the data by Dynamic Substitution.



For each data byte enciphered, the deg-9689 main RNG produces a 32-bit value. Eight RNG bits are used in the first-level Dynamic Substitution combiner. Four RNG bits select one from among the sixteen second-level Dynamic Substitution combiners. Eight RNG bits are used in the selected second-level combiner, which produces ciphertext.

Deciphering occurs in reverse order with inverse combiners.

Comments on Key Size and RNG Size

Perhaps the most controversial part of the Cloak2 design is the huge 992-bit size of the keys (the internal key and message key). Normally, an 80-bit key would be considered secure against a key-exhaustion attack. A birthday attack on 80-bit message keys could find some identical key values with about 2**40 messages; if this were important, perhaps we could argue that message keys should be 160 bits. So why are Cloak2 keys six times that length?

The main reason for huge keys is that they support a convenient, straightforward design at an acceptable cost. An extra 132 bytes per file seems negligible in the context of modern files, so -- in effect -- there is no reason **not** to use huge keys. The main expense in Cloak2 is the slight delay for RNG expansion to degree-9689, and this is not related to base key size. The expense involved in running an Additive RNG is independent of RNG size. There is of course some advantage to having a similar size for message key RNG init value, the message key itself, and the main RNG init value. There is also an advantage in using random message key values to directly seed the RNG, since this means that we avoid intermediate processing and so need not discuss the possible weaknesses of such processing. Huge keys also help to avoid the usual simplistic arguments based solely on key size, and so may contribute to a higher level of discussion of the design.

Even with a base state of 992 bits, we can fully seed only a relatively small RNG, and an Additive RNG of degree *n* can be completely penetrated with a knowledge of only *n* elements. But the attack must somehow produce *n* elements of information from the ciphertext, and in Cloak2 we only get one byte of ciphertext for each four-byte element from the RNG. Messages of under 38,756 bytes simply do not contain sufficient information to develop the state of a degree-9689 RNG. Stream-cipher messages with less state than the amount in the RNG are arguably absolutely secure.

The advantage of an RNG with huge state for messages containing more than that state is more ambiguous. Still, it is not completely unreasonable to expect that, if the isolation mechanisms do leak information, they may do so at some fraction of the data rate. This could greatly expand the size of the message such an RNG would arguably protect absolutely. Messages larger than that would be insecure at some cost level -- hopefully high, and repeated for every message. Of course, any cipher can only protect information which is otherwise secret, and virtually any other approach would be preferable to attacking Cloak2 ciphertext.

The step-by-step nonlinear expansion of the main RNG state through four buffers to degree 9689 is almost twice as expensive as a simple linear expansion in a single buffer. One could argue, however, that such linear expansion would allow one to attack the original degree-31 RNG state using the information in final RNG. Consequently, RNG expansion must be (and is) nonlinear.

Strength Arguments

Brute Force on the RNG

> The Opponent tries each possible RNG state until the cipher is broken.

There are 992 bits of independent RNG state: Clearly, it is only necessary to step through all possible states to "break" the cipher. This is the design strength. (Cloak2 is a secret key cipher and can be directly compared to the 56-bit keyspace of DES. That is, DES has 2**56 possible keys; Cloak2 has 2**992 possible keys.)

Brute Force on the User Key

> The Opponent tries each possible User Key until the message deciphers properly. Try most-likely keys first. (This is most applicable to the alias file key, which may be a language phrase.)

No cipher can do very much about this attack if a user really wants to use a weak key. The advanced Cloak2 program supports the automatic generation of random keys into an alias file, where they can be selected and used with a non-secret alias tag. This does not completely solve the problem, however, because the alias file itself requires a User Key.

Cloak2 supports User Keys of virtually arbitrary length, but even a long key is not necessarily "strong." Concatenating the names of family members will not make a good key. A strong key must be unique and unsearchable, and never written down or stored as plaintext in other files. Cloak2 alone cannot assure this.

Since the alias file User Key must be entered, it can be stolen, and so should be changed periodically. This is easy to do by deciphering the alias file under the current key and then immediately enciphering under the new key. In most cases, it is not the cipher design but instead the alias-file key which is the weakest part of the system.

Chosen Key

> The Opponent tries various keys, adaptively, and compares the resulting ciphertext to the real ciphertext, to try and gain insight into the correct key value.

Given that CRC is used to generate the internal RNG base state from the User Key, it is hard to see how any particular bit arrangement could possibly be preferred, or occur unusually often. This is one big reason for using CRC to process the User Key, as opposed to, for example, a "cryptographic" hash which has no known strong mathematical basis.

Note that the resulting CRC state is only used to produce an arbitrary message key cipher value, which ciphers arbitrary message keys. It is hard to imagine how changing the User Key could possibly produce directed results in the main data RNG, for example. Undirected results are unlikely to be useful.

Differential Cryptanalysis

> Here the Opponent exploits known properties of particular known substitution tables or transformations. Repeated similar transformations on data may produce an uneven distribution of bit probabilities. This information can be used to effectively reduce the number of "rounds" in an iterated block cipher.

Differential Cryptanalysis does not appear to apply to this cipher, because all Cloak2 tables are "keyed." That is, all Cloak2 tables are unbiased arbitrary permutations created by an RNG initialized by a particular message key. Since proper shuffling algorithms are used, every table permutation is equally probable. This means that the particular table arrangements will be unknown to The Opponent, so the advantage of a knowing a particular table arrangement is lost.

It is certainly true that some table arrangements can be said to be "better" than others in terms of block cipher avalanche. But these "weak" tables may not be weak when used in stream-cipher processing. And Dynamic Substitution tables continue to change during processing, so any possible advantage is at best transient.

Ciphertext Only

> The Opponent accumulates a mass of ciphertext material and tries to find relationships within the data which expose successive levels of the mechanism, until the cipher is broken.

The data flowing through the Cloak2 cipher are extensively randomized by two stages of adaptive Dynamic Substitution and a substantial jitterized Additive RNG. It is hard to imagine how a statistical relationship could survive this.

Known Plaintext

> The Opponent somehow "obtain" some amount of plaintext and the corresponding ciphertext. Ordinarily, this attack is on a conventional stream cipher using exclusive-OR combining, which will immediately reveal the confusion sequence. The confusion sequence, and knowledge of the RNG design is then used to develop the original cipher state, which breaks the cipher.

Cloak2 has a two-level combiner which combines a single byte of data with two bytes and four bits of confusion to produce a single byte of ciphertext. Accordingly, a single known-plaintext byte cannot, by itself, describe the confusion which produced the ciphertext.

Nor do single bytes reveal the Dynamic Substitution table states, because the selected table element (and indeed -- at the second level -- the table itself) changes at random after use. In contrast to other stream-cipher designs, known-plaintext will not immediately resolve the confusion value in a Dynamic Substitution design so that the RNG can be attacked.

Key-Stream Repetition

> The Opponent uses a known-plaintext to develop the values from the RNG, then waits for the RNG to cycle, to use those RNG values again.

Cloak2 uses RNG's with cycle lengths far too large to repeat in practice.

Birthday Attack on Message Keys

> The Opponent accumulates enough ciphertext to find two or more messages enciphered with the same message key, then tries to analyze the initial ciphering state, which must be the same if the message key is the same.

First, identical Cloak2 message keys can only be identified under the same User Key. That is, if the same message key value occurs under different User Keys, it will produce two different enciphered message key values. Thus, we will not know, externally, that the message keys are in fact the same.

Next, if we assume that message key values are "random like," the probability of finding two messages with the same 992-bit message key is very, very, very small. We would expect to find just two messages with the same message key after we have accumulated about 2**496 messages. This is essentially impossible.

Last, note that the Dynamic Substitution combiner state will deviate with each data difference. As the combiner states diverge, analysis of the RNG state would seem to be increasingly complex.

Chosen Plaintext

> The Opponent somehow arranges to use the cipher at will, but does not know the key. The cipher is to encipher potentially huge amounts of data specifically -- and sometimes dynamically -- chosen to reveal the cipher.

(It is not clear that this would be a realistic attack in the context of normal secret-key end-user-cipher operation: If The Opponent has access to the ciphering computer, the compute state probably could be stored to a file, which would then contain the internal key. This is more a "black bag" attack than cryptanalysis, but would far easier and thus far more likely under the stated conditions.)

Since message-key values are not under the control of the user, there simply is no fixed internal state to be examined by using different messages. And since both the Dynamic Substitution and RNG state is dynamic, there is no fixed internal state that can be examined in one message either.

The obvious attack on a single level of Dynamic Substitution is to use defined-plaintext repeatedly to traverse the entire table both before and then after a particular character is enciphered. This will reveal the confusion byte at the combiner so that the RNG can be attacked in the way of the usual exclusive-OR stream cipher. (Note that this requires far more work than the exclusive-OR combiner.)

Cloak2 complicates chosen-plaintext attacks in several ways:

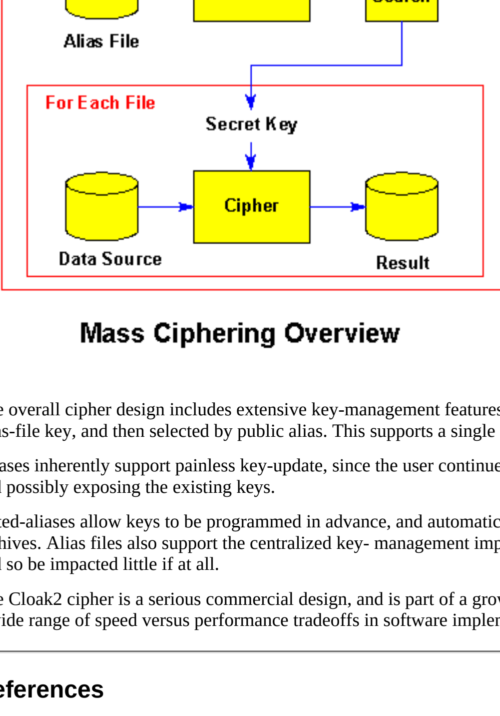
1. the RNG is large, and attacks on Additive RNG's are probably cubic in RNG length;
2. only 20 out of 32 of the RNG bits are available even assuming complete combiner penetration;
3. the RNG is isolated from attack by the nonlinear "jitterizer" mechanism;
4. the combiner is two levels deep, making the traversal attack complex; and
5. a defined-plaintext attack is not particularly useful in the context of a message key which is different for each and every message, since the RNG and combiner tables will be set up differently each time.

There is no known attack -- not even a theoretical attack -- on a dual-level-nonlinear-combiner message-key cipher like Cloak2.

Current Implementation

The current Cloak2 implementation is a relatively small (49K including substantial "help" information), relatively fast (154 KBytes/sec max. on a 486/DX2-50), command-line program for DOS on IBM PC-type machines. The current design includes an interactive command-line entry mode which is easy to use under Microsoft Windows. Inner ciphering loops are written in hand- optimized assembly-language for best speed, but the program will run on an old 8088 machine, and requires no numeric co-processor.

Cloak2 directly supports wildcard mass ciphering, enciphered secret key "alias" files, and enciphered Cloak2 "batch" files of Cloak2 commands:



The overall cipher design includes extensive key-management features to help create, transport, and use large random keys. Enciphered alias files allow many secret keys to be kept under a single alias-file key, and then selected by public alias. This supports a single key-entry by the user, since, if the key is entered incorrectly, the alias will not be found and ciphering will not occur.

Aliases inherently support painless key-update, since the user continues to use exactly the same alias for each new key. New Cloak2 keys can be added to the enciphered alias file without deciphering and possibly exposing the existing keys.

Dated-aliases allow keys to be programmed in advance, and automatically used when the date changes. The ability to access old, replaced keys by date supports access to old messages and entire archives. Alias files also support the centralized key- management important for business. As people leave, alias files can be extended with new keys, and ordinary users can use the exact same aliases and so be impacted little if at all.

The Cloak2 cipher is a serious commercial design, and is part of a growing family of serious ciphers and cipher engines for software developers. These designs use a wide range of technology to achieve a wide range of speed versus performance tradeoffs in software implementation.

References

[1] Ritter, T. 1990. Substitution Cipher with Pseudo-Random Shuffling: The Dynamic Substitution Combiner. *Cryptologia*. 14(4): 289-303.

[2] Ritter, T. 1990. Dynamic Substitution Combiner and Extractor. *U.S. Patent 4,979,832*.

[3] Ritter, T. 1991. The Efficient Generation of Cryptographic Confusion Sequences. *Cryptologia*. 15(2): 81-139.

Terry Ritter, his current address, and his top page.

Last updated: 1996-05-31

United States Patent 4,979,832

Dynamic Substitution Combiner and Extractor

Terry Ritter

A reversible nonlinear stream cipher combiner with state. One of the few new stream cipher combiners in the past 50 years.

For informational purposes only. Taken from the original Application and Amendment files which were sent to the PTO in printed form. At the PTO, the documents were manually transcribed into the printed patent. Here, the files were converted to HTML, partly by hand. For these reasons this version may differ, somewhat, from the printed patent. If there is any question, see an actual printed patent copy from the PTO, with the Certificate of Correction.

Contents

- Title
- References
- Abstract
- Drawings
- Body
  - Technical Field
  - Background -- Classical Prior Art
  - Background -- Current Prior Art
  - Background -- U.S. Pat. No. 4,195,196
  - Background -- U.S. Pat. No. 4,751,733
  - Description of Drawings
  - List of Reference Numerals
  - Description -- Fig. 1
  - Operation
  - Dynamic Substitution and Pseudo-Random Results
  - Dynamic Substitution in General
  - Other Embodiments
  - Multiple Realizations
  - Ramifications
  - Summary
  - Conclusion and Scope of Invention
- Claims

United States Patent

Ritter | **4,979,832**
Dec. 25, 1990

DYNAMIC SUBSTITUTION COMBINER AND EXTRACTOR

Inventors: **Ritter; Terry F.** (2609 Choctaw Trail, Austin, TX 78745).

Appl. No.: **431,016**
Filed: **Nov. 1, 1989**

Intl. Cl.: H04L 9/00
U.S. Cl.: **380/ 28**; 380/ 46; 380/ 49
Field of Search: 380/42, 28, 56, 57, 58, 46, 49

References Cited

U.S. PATENT DOCUMENTS

Re30,957	Jun., 1982	Feistel	380/ 42
1,310,719	Jul., 1919	Vernam	
2,496,317	Feb., 1950	Smith	
2,567,214	Sept., 1951	Kohler	
3,159,712	Dec., 1964	Arko	
4,202,051	May, 1980	Davida	375/ 2
4,751,733	Jun., 1988	Delayaye et al.	380/ 42
4,797,922	Jan., 1989	Massey	380/ 46

OTHER PUBLICATIONS

Knuth, The Art of Computer Programming, vol. II, pp. 31-32, (The MacLaren-Marsaglia Randomizer).

Michener, "The Generalized Rotor . . . ", Cryptologia, Apr. 1985, pp. 97-102.

Rubin, "Foiling an Exhaustive Key-Search Attack", Cryptologia, Apr. 1987, pp. 102-104.

Algorithm 235, Random Permutation, Procedure Shuffle, R. Durstenfeld, Communications of the ACM, vol. 7, No. 7, Jul. 1964, p. 420.

Primary Examiner: Tarcza; Thomas H.
Assistant Examiner: Cain; David

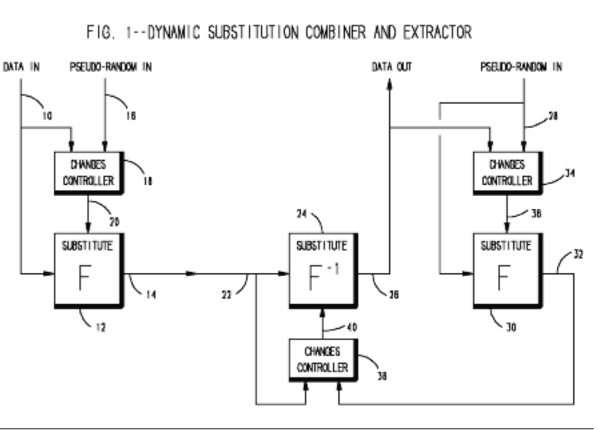
ABSTRACT

A first data source and a second data source are combined into a complex intermediate form or result; the first data source can be extracted from the intermediate form using the second data source. The combining mechanism can provide an alternative to the exclusive-OR combiner commonly used in stream ciphers.

Each data value from the first data source is transformed by substitution using one of potentially multiple translation tables (**12**). The translations within each table can be changed after each substitution operation using a changes controller (**18**). Commonly, the just-used table is re-arranged or permuted; permutation retains invertibility, so that the ciphertext may be deciphered. As a particular design, the just-used substitution element may be exchanged with some element within the same table, as selected by the second data source, after every translation.

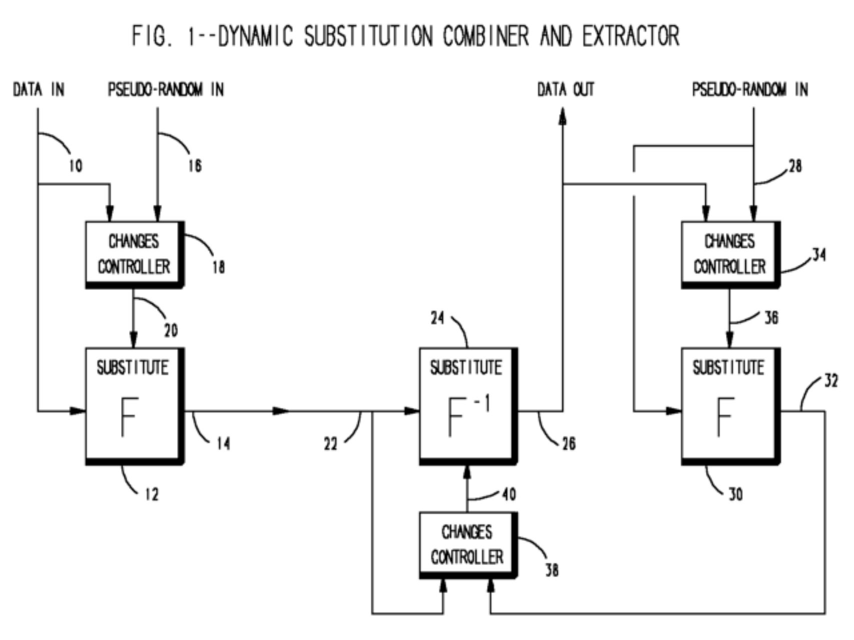
The combiner substitution tables (**12**) can easily be forced to be invertible, and when they are, extraction is possible by substitution through inverse tables (**24**). Valid inverse tables can be maintained provided that the same second data is used by both mechanisms. This is the normal situation in stream ciphers.

The combiner can also be used to combine two pseudo-random confusion streams into a more-complex confusion stream. In this case, extraction may be unnecessary and so the combiner substitution tables need not be invertible. Thus, the translation changes need not be limited to permutations.



19 Claims, 1 Drawing Sheet

DRAWINGS



BODY

DYNAMIC SUBSTITUTION COMBINER AND EXTRACTOR

TECHNICAL FIELD

This invention relates to the art of cryptography, or the secure transmission of information. More particularly, this invention relates to mechanisms or processes for combining two data sources into a result, and to inverse mechanisms or processes for extracting data from a combined result.

BACKGROUND -- CLASSICAL PRIOR ART

For the purposes of this patent, the term mechanism will refer to the logical concept of a machine, which may be realized either as a physical machine, or as a sequence of logical commands executed by a physical machine.

The object of cryptography is to transform information or plaintext into an intermediate form or ciphertext which conceals the information, but which may be used to reproduce the information at a different place or later time. The main problem of cryptography is the construction of intermediate information forms which can be efficiently created and then used, given a special understanding or key, but which are extremely difficult to use without that special understanding.

This invention may be used to translate plaintext data into ciphertext using a key, and to recover the original plaintext data using the same key. A particular key value would typically select the generation of a particular random-like or pseudo-random sequence of confusion data; the design of such a generator is well understood and not part of this invention. The confusion data would be combined with the plaintext data in order to encipher or hide the plaintext. The same confusion data would be inversely combined with the ciphertext data in order to decipher or recover the original plaintext. Various other applications are also possible.

One of the ancient forms of cipher is known as substitution. In a monoalphabetic substitution cipher, each plaintext letter has one fixed ciphertext substitute. In this case, the key is the set of substitutions, or substitution alphabet which is used. One way to implement substitution is through the use of a translation table. Conceptually, a translation table is just a list of the output or substitute values, listed in the order of the input or to-be-substituted values. Substitution consists of using the input value to select the particular substitute value or translation which is the result. Each of the possible substitutions in a single table may also be called a substitution element.

All ciphers must, at least in principle, confront an "enemy" or opponent who seeks the information contained in the ciphertext. Substitution is generally easy to solve or penetrate because the different letters in a written language are used with different characteristic frequencies, and substitution does not change those relationships. Accordingly, a cryptanalyst attempting to penetrate the cipher can often make good guesses for various letter substitutions. And, since language is naturally redundant, those guesses generally can be extended with other guesses until the message becomes fully known.

In a polyalphabetic substitution cipher, each letter has multiple fixed substitutes; that is, multiple substitution alphabets are used in some sequence. The different substitution alphabets help to obscure the letter-frequency statistics of the plaintext, but the fixed nature of the substitution alphabets, and their use in a fixed sequence, still allows fairly easy penetration, given a reasonable amount of ciphertext. Although small substitutions are still used, substitution has largely been abandoned as the central part of serious ciphers because it is so easily penetrated.

BACKGROUND -- CURRENT PRIOR ART

The current state of the art of cryptography includes stream cipher systems. Stream ciphers operate on a stream, or sequence, of plaintext data, one element at a time, to produce a stream of ciphertext.

A stream cipher generally combines plaintext data with pseudo-random confusion data to produce ciphertext data. A combining function or combiner mechanism in some way mixes two data sources to produce a typically more complex result. That is, many stream ciphers are grossly just a pseudo-random sequence generator and a combiner (as one example of many, see U.S. Patent 4,202,051 issued May 6, 1980 to G. Davida and D. Wells). The combining function generally used is a simple binary bit-by-bit addition mod 2, which is also known as the Boolean logic exclusive-OR function. One form of exclusive-OR is a two-input one-output logic function which is now commonly available as an integrated circuit electronic digital logic building-block. The same function is also available as a logical command on most general-purpose computers or microprocessors.

The exclusive-OR function has been used for stream-cipher data encryption for more than 70 years (the first known version, using electro-mechanical relay logic, can be seen in U.S. Patent 1,310,719 issued July 22, 1919 to G. Vernam). Various embodiments of the exclusive-OR combiner function have been patented (for example, see the electronic vacuum tube versions in U.S. Patent 2,496,317 issued Feb. 7, 1950 to H. Smith; U.S. Patent 2,567,214 issued Sept. 11, 1951 H. Kohler; and a mechanical version in U.S. Patent 3,159,712 issued Dec. 1, 1964 to R. Arko). The exclusive-OR combiner is useful for encryption because it helps to disguise the frequency statistics of the plaintext data; if either input to the exclusive-OR combiner is a pseudo-random sequence, the result will generally have random characteristics. In the case of exclusive-OR combiners, the exact same mechanism used for combining the plaintext data into ciphertext can also be used to extract the plaintext data from the ciphertext. In contrast, the combining mechanism of this invention requires a different but related mechanism to extract the plaintext data.

On the surface, the use of the exclusive-OR function as a cryptographic combiner apparently reduces the ciphering problem to the design of a confusion generator or random number generator which would be exceedingly difficult for a cryptanalyst to fully analyze. Accordingly, we see the continuing development of ever more complex random number generators (for example, see the above mentioned U.S. Patent 4,202,051 issued May 6, 1980 to G. Davida and D. Wells). But the exclusive-OR combiner itself has properties which actually aid in the cryptanalysis of the confusion stream. For example, if an analyst is able to obtain some amount of plaintext plus the matching ciphertext, the analyst can recover that portion of the raw confusion source. Normally we think of an analyst trying to decipher the encrypted data, but if the analyst can fully analyze and reproduce the pseudo-random confusion source, all subsequent messages can be deciphered. Such analysis may even be possible under the assumption that some particular words will appear at particular places in a message, as typically occurs in standard message, memo, and business letter formats.

The obvious alternative of selecting some other simple Boolean logic function to replace the exclusive-OR combiner does not work. The other two-input Boolean logic functions (e.g., AND, OR) are not suitable; if these functions have a pseudo-random sequence on one input, the output is generally not random-like. More complex combining functions are the current state-of-the-art (for example, see U.S. Patent 4,797,922 issued Jan. 10, 1989 to J. Massey and R. Rueppel). Note that Massey's device simply combines confusion streams, and so need not be reversible; in contrast, a combiner for data and confusion would have to be reversible so that the enciphered data could be deciphered.

BACKGROUND--U.S. PATENT 4,195,196

Another example of the prior art occurs in U.S. Patent 4,195,196 reissued 1 Jun 1982 (Re. 30,957) to H. Feistel. This mechanism can be seen as a stream cipher system which is a product cipher or multiple-level cipher, each level consisting of an exclusive-OR combiner and a "confusion" stream. Multiple pseudo-random confusion streams are generated simultaneously by transformations of stored state and the plaintext and ciphertext data from each level; thus, this mechanism is an example of an autokey cipher, in which the confusion data is generated as a complex transformation of the key, initialization, and message data.

Each cipher level of the mechanism consists of an exclusive-OR combiner and single modified autokey confusion sequence. However, due to the mathematical characteristics of the exclusive-OR operation, the ciphertext result will not sensibly differ from a single exclusive-OR combiner and single modified autokey confusion generator which holds several internal delayed combinations of message data. This is important, because it reveals that the mechanism may be vulnerable to the "known plaintext" attack discussed earlier.

If plaintext data are exclusive-OR combined with the associated ciphertext data, the result is a confusion sequence which can be directly related to a modified confusion generator. A cryptanalyst could use the re-generated confusion sequence and plaintext message data to attempt to define the internal state and key in the modified generator. If successful, this would penetrate the modified confusion generator, and also the equivalent cipher system using the un-modified confusion generator.

Thus we see that this mechanism is an example of a classical Vernam stream cipher: a pseudo-random confusion generator plus conventional exclusive-OR combining. It is thus susceptible to the plaintext attack, which is a weakness of all exclusive-OR combiners. A similar system, one using exactly the same confusion generator but with a combiner less susceptible to attack, would seem to be a significant improvement.

BACKGROUND--U.S. PATENT 4,751,733

Yet another example of prior art occurs in U.S. Patent 4,751,733 issued 14 Jun 1988 to B. Delayaye and A. Lebret. This is a substitution-permutation product cipher which applies a sequence of substitutions and permutations to the plaintext data, with each substitution selected by the key from a set of pre-defined tables.

In some sense the substitution tables are "changed" under different keys, since each key selects a particular group of active tables. However, the key does not change during ciphering, and so the ultimate transformation, from input to output, while complex, also does not change during ciphering, and this is a weakness. The preferred embodiment is a 32-bit transformation from input to output; in theory, if the mechanism were left unguarded, the complete 32-bit to 32-bit transformation could be investigated and recorded, and the system thus penetrated until the pre-defined substitution tables were eventually updated and changed.

Even if the key were "randomized" or otherwise "stepped" during operation (a possibility seemingly not supported by the patent), the mechanism would still be restricted to selecting complete tables from a fixed set of unchanging substitution tables.

Because static pre-defined tables constitute the heart of this mechanism, it is essential that they be retained as secrets. Since the table contents do not change, they are amenable to cryptanalysis and eventual penetration. Consequently, the tables will need to be changed on some regular basis, thus initiating an industry in which the tables must be created, manufactured, stored, transported, and retained, all in absolute secrecy. It would seem that an improved combiner mechanism, one less susceptible to cryptanalysis and requiring less of a support industry, must necessarily yield a big improvement in this cipher system.

OBJECTS AND ADVANTAGES

One of the objects of this invention is to provide a mechanism or process by which plaintext or other data can be combined using confusion data to produce a complex intermediate form or ciphertext result. The mechanism of this invention thus provides the basis for a cryptographic system.

Another object of this invention is to provide a combining mechanism or process which can create a multitude of intermediate forms depending on the combination of plaintext and confusion values. This means that attempts to penetrate the ciphertext by trying every possible intermediate form are unlikely to succeed.

Yet another object of this invention is to provide a combining mechanism or process which is more difficult to cryptanalyze than a simple exclusive-OR combining function. This is a significant advantage, because such an analysis could penetrate the system for all future messages.

Another object of this invention is to provide a combining mechanism which will hide the confusion data when the cryptanalyst has both some amount of plaintext as well as the associated ciphertext. Since it is difficult to prove that any particular pseudo- random confusion generator will not ever be cryptanalyzed, hiding the confusion data and thus complicating such an analysis is a big advantage.

Still another object of this invention is to provide an efficient inverse mechanism or process by which previously-combined data can be separated or extracted, using the confusion data involved in the original combination. Since deciphering is normally required, an efficient extraction mechanism can make the whole system practical.

Another object of this invention is to provide a mechanism or process by which two confusion sources can be combined to produce a more-complex confusion result which may be used by some other combiner mechanism. Since the analysis of the confusion data could lead to penetration of a cipher system, the ability to create ever more complex confusion data is a big advantage.

And yet another object of this invention is to provide a form of substitution which tends to randomize the letter-frequency statistics of the plaintext data. Since the conventional statistical attacks on substitution utilize these statistics, those attacks would tend to be rendered ineffective.

And still another object of this invention is to provide an encryption module which can be used as part of a complex encryption system composed of a network of cryptographic modules, each potentially different. This would allow the encryption system to vary from message to message, or even dynamically within a message. Since the encryption system itself could change, it would be hard for a cryptanalyst to know how to start working on the other aspects of the system.

Further objects and advantages of my invention will become apparent from a consideration of the drawing and the ensuing description.

DESCRIPTION OF DRAWINGS

FIG. 1 shows a combining mechanism, for enciphering data, and also the required inverse or extracting mechanism, for deciphering the enciphered data.

LIST OF REFERENCE NUMERALS

10: combiner substitution input
12: combiner substitution
14: combiner output
16: combiner substitution changes input
18: combiner substitution changes controller
20: combiner substitution changes controls
22: extractor inverse substitution input
24: extractor inverse substitution
26: extractor output
28: extractor substitution changes input
30: extractor substitution
32: extractor substitution output
34: extractor substitution changes controller
36: extractor substitution changes controls
38: extractor inverse substitution changes controller
40: extractor inverse substitution changes controls

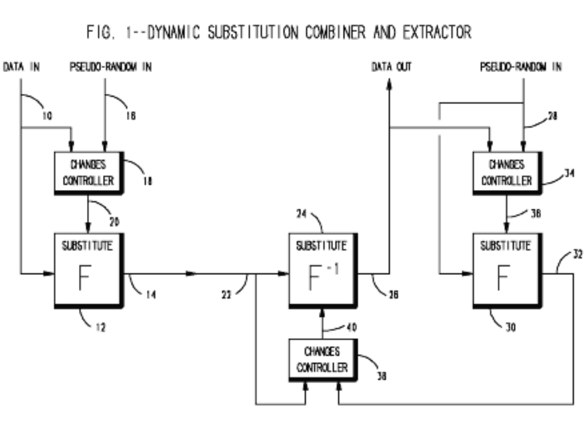
DESCRIPTION -- FIG. 1

A typical embodiment of any cipher system would include both enciphering and deciphering; thus, **FIG. 1** consists of a combiner or enciphering section and an extractor or deciphering section. As in other stream ciphers, it is assumed that identical pseudo- random sequences are available for enciphering and deciphering. **FIG. 1** is the currently preferred embodiment, in which the substitution alphabets are changed under the control of two values. In this embodiment, the substitution elements selected by these values are exchanged. Other embodiments might require other or additional change- selection inputs, and might make more changes in the substitution block than a simple exchange of two elements.

FIG. 1 is a block diagram of the type commonly employed by those skilled in the art of electronic design. The **FIGURE** may be interpreted as a network of simultaneous independent functions which receive data, process it, then pass the data on to the next function. The same **FIGURE** may also be interpreted as a time sequence of different operations on a single block of data which is not moved. Presumably there are also other alternate interpretations. In truth, the particular design selected for a real system will depend on engineering trade-offs which are at the heart of the normal practice of the art of systems design.

In the combiner section, combiner substitution **12** translates combiner substitution input **10** data into combiner output **14** data. The result would be a simple substitution, except that the substitution **12** will change. A substitution or inverse substitution would typically be implemented as addressable storage, and realized with an electronic memory device, or an addressable area of memory hardware in an electronic digital computer or microprocessor.

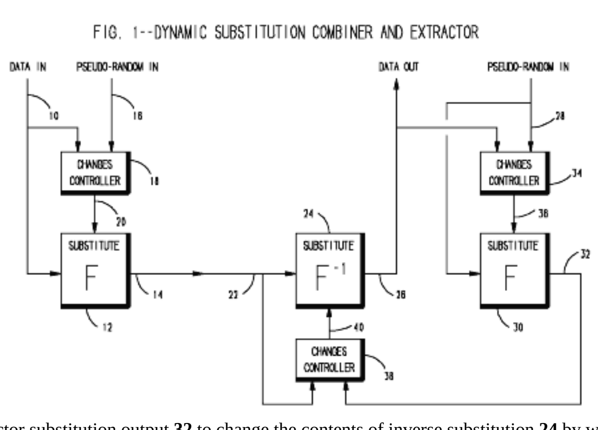
The substitution changes controller **18** uses both substitution input **10** and combiner substitution changes input **16** to change the contents of substitution **12** by way of combiner substitution changes controls **20**. Typically, substitution input **10** and changes input **16** each select an address or substitution element in substitution **12**; the selected substitution elements are then exchanged.



The particular type of changes controller required in a particular design would typically be implemented in electronic logic by someone skilled in the art of digital electronic design. Alternately, the changes could be implemented as a sequence of instructions to an electronic digital computer or microprocessor to change the values in the substitution area of memory hardware. Since, in this preferred embodiment, the controller always exchanges exactly two elements stored in a memory device, this would be a fairly basic and straightforward design.

The combiner output **14** is connected to the extractor input **22** so that deciphering may occur. This connection represents data transmission and/or storage.

In the extractor section, extractor inverse substitution **24** translates extractor inverse substitution input **22** data into extractor output **26** data. This would also be a simple substitution, except that this substitution is also changed; the resulting data out **26** is the same as the combiner data in **10**. The extractor substitution **30** translates the extractor substitution changes input **28** data into the extractor substitution output **32** data; extractor substitution **30** is the same as combiner substitution **12**, and is used to develop translated pseudo-random data to change inverse substitution **24**.



The extractor changes controller **34** uses both the extractor output **26** and the extractor substitution changes input **28** to change the contents of the substitution **30** by way of extractor substitution changes controls **36**. In this way, substitution **30** in the extractor is kept up with the identical substitution **12** in the combiner. This occurs because the correctly-deciphered Data Out value (on output **26**) is the same as the Data In value (on input **10**) which was enciphered; the same pseudo-random sequence is also used by both units, so whatever changes occur in combiner substitution **12** can be made to occur identically in extractor substitution **30**.

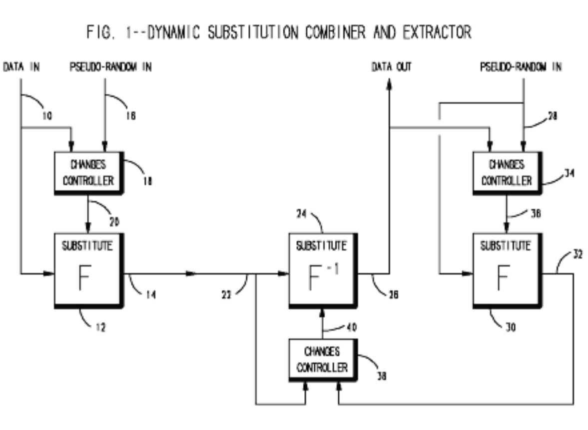
The extractor inverse changes controller **38** uses both extractor input **22** and extractor substitution output **32** to change the contents of inverse substitution **24** by way of the extractor substitution changes controls **40**. Since plaintext data and pseudo-random data are used to change substitution **12**, substituted plaintext data and substituted pseudo-random data are appropriate to change inverse substitution **24**. In this way, the deciphering inverse substitution **24** keeps up with the changing enciphering substitution **12**.

OPERATION

A plaintext value on input **10** is transformed by substitution **12** into a ciphertext value on output **14**. A ciphertext value on input **22** is transformed by substitution **24** into the original plaintext value on output **26**. In order for this to work, substitution **12** must be invertible.

Substitution **12** will be invertible if it contains no more than one instance of any particular output value. In practice this is easy to guarantee. For example, the substitution table can be made exactly as large as the number of possible input values **10**, and filled sequentially with the possible output values. If no output value appears more than once, substitution **12** will be invertible. Substitution **12** can then be shuffled or randomized in any number of ways; as long as the values in the table are simply re-arranged or permuted, substitution **12** will remain invertible.

After substituting, some of the substitution elements in substitution **12** will be re-arranged. In order to decipher subsequent data, inverse substitution **24** must also be re-arranged, but in an inverse manner. To make this easier, an extractor can maintain its own substitution **30** which tracks substitution **12** in the combiner.



In the preferred embodiment, controller **18** exchanges two substitution elements; these elements are selected by the plaintext value **10** and the pseudo-random value **16**. These same values will also be available to controller **34** in the extractor, to change substitution **30** in exactly the same way. The plaintext input **10** is available because it was enciphered and then deciphered as the extractor output **26**. The changes inputs, on the combiner **16** and the extractor **28**, must be exactly the same pseudo-random sequence. This is a normal requirement for a stream cipher, and also a requirement of the combining mechanism of this invention.

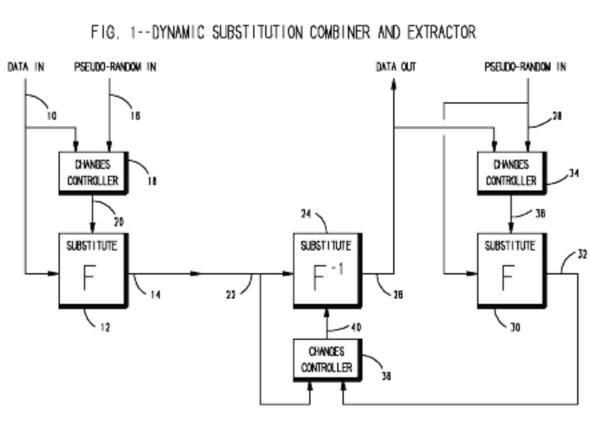
But before substitution **30** is changed, it is used to translate the pseudo-random value on input **28** into a value on output **32**, which selects one of the elements which must be changed in substitution **24**. The other element which must be changed is selected by the ciphertext value on the extractor input **22**. These values are the enciphered or translated versions of the Data In (on input **10**) and Pseudo-Random In (on input **16**) values which are used to change substitutions **12** and **30**. The translated values select the correct elements to be exchanged, because the entire inverse substitution **24** is simply a translation of the combiner substitution **22**. Controller **38** exchanges the two elements in the inverse substitution **24** as selected by the translated values.

DYNAMIC SUBSTITUTION AND PSEUDO-RANDOM RESULTS

It is interesting, important, and unexpected, that if either combiner input has random characteristics, the output will also have random characteristics, as long as the input streams are not correlated. There are two cases:

Case 1: First, we assume a random input **10**, a constant input **16**, and that the substitution elements in **12** are evenly distributed. We note that the substitution elements in **12** will be evenly distributed if the overall substitution **12** is invertible, and it generally will be invertible to support deciphering. When the substitution elements are evenly distributed, any input distribution must produce a similar result distribution, so a random input **10** will produce a random output **14**.

Case 2: Next, we assume a random input **16**, and a constant input **10**. The value on input **10** will select a particular substitution element in **12** to be sent to output **14**. But each time this occurs, the value of that particular element will be exchanged with the value of some element selected by input **16**, which is random. Consequently, the value of the particular element selected by constant value **10** changes at random after each substitution, so the output value **14** is random, despite a constant input **10**.



Naturally, if the sequences on input **10** and input **16** were similar or correlated, the combiner could be made to produce non-random output (an exclusive-OR combiner could be defeated similarly). But this is normally made very unlikely by cryptographic design.

Thus, a random sequence on either input of the combiner of this invention will generally produce a random sequence out. This is similar to the statistical characteristics of the exclusive-OR combiner. These unexpected and advantageous results seem to make this invention statistically comparable to the conventional exclusive-OR combiner. But this invention does not have the cryptanalytic weakness of the exclusive-OR combiner, and thus would seem to be superior overall.

DYNAMIC SUBSTITUTION IN GENERAL

Plaintext data values are processed sequentially, typically character-by-character. Each plaintext character value selects an element in a substitution table, and that element becomes the ciphertext value. After each substitution, the values in the substitution table may be changed or re-organized. In the preferred embodiment, the just-used substitution value is exchanged with some value in the table selected by another data sequence; commonly, this other sequence will be pseudo-random.

If the substitution table is invertible, any particular ciphertext value may be translated back into plaintext with a suitable inverse substitution table. However, when the forward substitution table changes, the inverse substitution table must also change. In the preferred embodiment, the change in the forward table is a simple exchange of two elements; thus, two elements must also be exchanged in the inverse table. Both of the elements to change in the inverse table are the enciphered values of the elements exchanged in the forward table. One of the elements is the enciphered plaintext value, which is the ciphertext value, and is directly available. The other element is the enciphered pseudo-random value, and this is not directly available.

In order to identify the second element to be exchanged in the inverse table, it is reasonable to maintain a forward substitution table. This table can be maintained directly, since both the plaintext value (as deciphered) and the pseudo-random value (reproduced locally) are available. Given a forward substitution table, the pseudo- random value can be enciphered to identify the second element to be exchanged in the inverse table.

A polyalphabetic version, with multiple invertible tables in each case, is likely to be even more secure, at the expense of some extra storage and a wider-range pseudo- random sequence.

OTHER EMBODIMENTS

The discussion of **FIG. 1** described a particular currently-preferred embodiment of this invention, but many other embodiments are possible. For example, many different control circuits could be designed by a practitioner skilled in the art of electronic digital design, and each of the designs might still perform the same functions needed in this invention.

As examples of other possible embodiments, any desired amount of substitution element permutation could occur, after each translation, or perhaps after several or a variable number of translations. A wider range of data values could be accommodated by expanding data paths and storage cells in the substitution memories. Moreover, various designs could choose to do the various operations in different sequences, or in parallel. It is well within the responsibilities of an implementor skilled in the art of electronic design to select the sequences in which functions will be performed, and even those functions which can be performed in parallel.

FIG. 1 and the description deal with particular functional blocks, but the same processes could be implemented with different blocks. The functional decomposition of a concept into separate modules or blocks for a particular design is a standard task for a skilled practitioner of any system design technology. The design variations of this invention are essentially endless.

MULTIPLE REALIZATIONS

Virtually any logical computation technology could be used to realize this invention. Such technologies might include electronics, microelectronics, mechanics, light, fluidics, chemical or biological systems, etc. The invention could be realized as a sequence of instructions executed by a general purpose computer or microprocessor; it could be realized as software or microcode or as part of a custom microprocessor or integrated circuit device. Various hardware or software realizations could do the individual operations in different sequences, or in parallel, or combine hardware and software operations. Moreover, a complex system might include this invention as but a small part of its overall operations.

RAMIFICATIONS

The discussion of **FIG. 1** described a particularly practical use of the combiner. This use was as a cryptographic system, which combined plaintext data and an external pseudo-random sequence into ciphertext data. The original plaintext data could then be recovered by extracting the ciphertext data with the same pseudo-random sequence. Presumably, the pseudo-random sequence used for combining would be initialized by some sort of key value, and the exact same sequence could be developed for extraction only by using the exact same key.

Another use for a dynamic substitution combiner would be to combine two different pseudo-random sources. This would generate a more-complex pseudo-random combination, and would also help protect both input sources from analysis better than the simple exclusive-OR combiner generally used. In this case, an extractor would generally be unnecessary, since the same combined result could be reproduced by generating the original pseudo-random sources and combining them.

Yet another application for the mechanisms of this invention would be as modules or building-blocks of a large complex ciphering system with multiple different module designs. Each module could produce key-controlled results. Variously-constructed data could flow from module to module to generate an eventual result. The arrangement of ciphering modules might be fixed by design, or selected by the key, or might even change dynamically as a result of a pseudo-random sequence, or even cipher-network values.

Because the combiner can generate more-random data, or selectively randomized data, it might well have application in data transmission and data modulation (e.g., modems), radar, sonar, data storage, data conversion, facsimile, graphics, numerical statistics, simulation, artificial life, and artificial intelligence; indeed, any field which benefits from the use of pseudo-random sequences.

SUMMARY

Thus the reader will see that the combiner herein described can generate a continually-changing yet reversible combination of two data sources which is an improvement over the simple exclusive-OR combiner used in most stream ciphers.

The method of the present development is to revive and generalize the venerable substitution cipher, with the addition of a new data input which is used, along with the substitution input, to alter the contents of the translation table during operation.

The new data input, generally used for a confusion source, turns the ancient substitution operation into a combining function, with many new possibilities for interconnection and use.

Analysis of this structure and experimental implementations have shown that it can obscure the normal usage-frequency statistics. Whenever a data value is substituted, that particular substitution becomes known, at least potentially, to an external cryptanalyst. But, as soon as that substitution is used, it is changed, and the cryptanalyst has no way to know what it has been changed to.

The same mechanism can function with either data or confusion values on either input, depending on the goals of the designer. Two confusion sources might be combined to make a more complex result, and even two data sources might be combined for some reason.

If the combiner substitution is made invertible (that is, containing no duplicated values), it may be changed by re-ordering (permutation) in any way and still remain invertible. Thus, an inverse substitution can be used to decipher data enciphered by the combiner, and can keep up with the changing substitution by changing in an inverse way.

But if the combiner result need not be deciphered, as in the case of combining two confusion sources into a more-complex result, then no inverse substitution is necessary.

An efficient polyalphabetic version of the invertible combiner is possible using multiple substitution blocks.

This invention improves upon the security of a classical substitution cipher by essentially using a different substitution alphabet for every data value enciphered.

CONCLUSION AND SCOPE OF INVENTION

This invention extends cryptographic substitution by allowing the substitution tables to change dynamically. It is important and also unexpected that this mechanism generally prevents plaintext letter frequency information from appearing in the ciphertext. Since letter frequency information is generally sufficient to break a substitution cipher, this is a big advantage.

This invention extends cryptographic substitution by teaching a substitution mechanism to translate plaintext data into a multitude of different ciphertext intermediate forms, based on a second data sequence. This sequence would typically be pseudo- random and initialized by some sort of key value. In this way, a multitude of different keys can be supported, so that the attack of trying every possible key can be made unreasonable.

This invention extends cryptographic substitution by adding a new input to control substitution changes. It is important that the added input converts a one-input substitution mechanism into a two-input combining mechanism with many more possibilities for interconnection and use.

This invention extends cryptographic combining by teaching a new combiner mechanism with statistical effects comparable to those of the conventional exclusive-OR combiner. Thus, the new mechanism would seem to fit easily into current cryptographic system designs.

This invention extends cryptographic combining by teaching a new combiner mechanism to complicate penetration to the pseudo-random sequence in the event that a cryptanalyst has both ciphertext and the corresponding plaintext. This is important because it is very difficult to design a pseudo-random sequence generator which cannot be penetrated and past experience would lead us to mistrust any such claim. And if the pseudo-random sequence is penetrated, the cipher is broken.

This invention extends cryptographic combining by teaching a mechanism to extract plaintext from the ciphertext produced by a combiner of this invention. It is important and unexpected that it is useful for the extractor to maintain a forward substitution identical with that used in the combiner, in order to efficiently maintain an inverse substitution.

While my above descriptions contain many specificities, these should not be construed as limitations to the scope of the invention, but rather as an exemplification of a preferred embodiment thereof. Many other variations are possible. Accordingly, the scope of the invention should be determined not by the embodiments illustrated, but by the appended claims and their legal equivalents.

CLAIMS

I claim as my invention:

1. A mechanism for combining a first data source and a second data source into result data, including:
(a) substitution means for translating values from said first data source into said result data or substitute values, and
(b) change means, at least responsive to some aspect of said second data source, for permuting or re-arranging a plurality of the translations or substitute values within said substitution means, potentially after every substitution operation.

2. The combining mechanism of claim **1** wherein said substitution means is initialized to contain at most one occurrence of any particular translation.

3. The combining mechanism of claim **1** wherein the translations in said substitution means are held in a plurality of translation tables or lists.

4. The combining mechanism of claim **3** wherein each translation table in said substitution means is initialized to contain at most one occurrence of any particular translation.

5. The combining mechanism of claim **3** wherein some aspect of said second data source is used to select a particular translation table to be used for substitution and permutation.

6. The combining mechanism of claim **3** wherein every just-used translation in a translation table is exchanged with some translation in the same table.

7. The combining mechanism of claim **1** wherein those translations in said substitution means which are permuted by said change means include each translation which has been selected by said first data source subsequent to a previous permutation of that translation.

8. The combining mechanism of claim **1** wherein every just-used translation in said substitution means is exchanged with some translation in said substitution means.

9. A mechanism for extracting original data from a first data source using a second data source, including:
(a) first substitution means for inverse translating values from said first data source into said original data or resulting data values,
(b) second substitution means for translating values from said second data source into translated second source values,
(c) second change means, at least responsive to some aspect of said resulting data values, for permuting or re-arranging a plurality of the translations within said second substitution means, and
(d) first change means, at least responsive to some aspect of said translated second data values, for permuting or re-arranging a plurality of the translations within said first substitution means.

10. The extracting mechanism of claim **9** wherein the translations in said first substitution means and said second substitution means are held in a plurality of translation tables.

11. The extracting mechanism of claim **10** wherein some aspect of said second data source is used to select the translation tables used for substitution and permutation.

12. The extracting mechanism of claim **10** wherein every just-used translation in a table is exchanged with some translation in the same table.

13. The extracting mechanism of claim **9** wherein said resulting data and said second data source are used to select two elements in said second substitution means, and said change means acts to exchange the values of the selected elements.

14. The extracting mechanism of claim **9** wherein said first data source and said translated second data source are used to select two elements in said first substitution means, and said first change means acts to exchange the values of the selected elements.

15. A two-input one-output logic mechanism or design, which combines a first input value with a second input value, including:
(a) substitution means, potentially including a plurality of storage means, for saving substitute values and translating said first input value into an output value, and
(b) change means, at least responsive to some aspect of said second input value, for re- defining a proper subset of the substitute values within said storage means, potentially after every substitution operation.

16. The logic mechanism of claim **15** wherein each storage means in said substitution means is initialized to contain at most one occurrence of each substitute value.

17. The logic mechanism of claim **15** wherein said change means alters substitute values in said substitution means only by permutation or re-arrangement within individual storage means.

18. The logic mechanism of claim **15** wherein said change means alters substitute values in said substitution means by the exchange of substitute values.

19. The logic mechanism of claim **15**, further including:
(c) another instance of the logic mechanism of claim **15**,

wherein the two instances of the mechanism together extract a result data value from a first data value using a second data value, by placing said first data value on said first input value input of the first mechanism, and using said output value from the first mechanism as said result data value, and connecting said result data value to said second input value input of the second mechanism, and placing said second data value on said first input value input of the second mechanism, and connecting said output value of the second mechanism to said second input value of the first mechanism.

Terry Ritter, his current address, and his top page.

Last updated: 1997-02-04

Dynamic Substitution in Stream Cipher Cryptography

A Reversible Nonlinear Combiner with Internal State

A *Ciphers By Ritter* Page

Terry Ritter

Dynamic Substitution is a replacement for the weak exclusive-OR combiner normally used in stream ciphers. Dynamic Substitution is generally nonlinear, with substantial internal state. Multiple nonlinear combining levels are practical, as are selections among multiple different combiners. The resulting strength can support the use of weaker but faster keying generators. Additional strength is available in "hardened" and customized versions.

Contents

- Introduction
- Structure
  - Substitution Tables, Changes Controller
- Operation
  - Enciphering, Deciphering
- Design Opportunities
  - Use Modern VLSI Strengths, Make Each Combiner Unique, Use Nonlinear Combiners, Use Multiple Combiners in Sequence, Use a Dynamic Selection Among Multiple Combiners
- Negatives
  - Size, Speed, Error-Expansion
- Strength
  - Indisputably Stronger than Exclusive-OR, Known Plaintext Attack, Defined Plaintext Attack
- Hardening Dynamic Substitution
  - Customization Opportunities
- Summary
- Acknowledgements

Introduction

Dynamic Substitution is a reversible nonlinear combiner based on substitution.

A *combiner* is a cryptographic mechanism which mixes a confusion or keying sequence with data. A combiner is the heart of a stream cipher, which generally uses an "additive" combiner such as exclusive-OR.

Unfortunately, additive combiners have absolutely no strength at all: If an Opponent somehow comes up with some plaintext and matching ciphertext, an additive combiner immediately reveals the confusion sequence. This allows the Opponent to begin work on breaking the confusion sequence generator.

In a similar *known-plaintext* situation, Dynamic Substitution *hides* the confusion sequence (to some extent). This strengthens the cipher and allows the use of a weaker (and faster) confusion sequence generator.

The sources of the Dynamic Substitution idea were:

- a need to improve the strength of a conventional stream cipher, and
- a desire to negate the attacks on simple substitution.

Attacks on simple substitution usually contain the implicit assumption that the substitution table is static or unchanging. One way to avoid these attacks is to change the contents of the table dynamically. The signal needed to control these changes makes simple substitution into a new type of combiner, thus solving both problems at once.

Structure

The Dynamic Substitution mechanism consists of one or more invertible substitution tables, and some way to change the arrangement of the values in the tables.

Substitution Tables

An invertible substitution table contains each possible value exactly once. For example, a "byte-wide" table will have 256 byte-value entries. Any entry in the table can be selected by a value in the range 0..255, and every possible substitution value 0..255 will occur exactly once.

Each substitution table is "keyed" or shuffled under the control of a keyed random number generator (RNG). The keying process is very conventional. However, in Dynamic Substitution, a form of shuffling continues during operation, and this is unique.

Changes Controller

One way to "harden" simple substitution is to re-shuffle the entire table after every character substituted. But what does this gain us? The only entry in the table which can possibly be revealed by a substitution operation is the particular entry which was just used. Accordingly, if we change the table after *each* substitution, the *only* entry in the table that we really need to change is the just-used entry.

What do we change the just-used entry to? Well, we want the new entry to be "unknown," that is, an even-probability selection from among all possible values. One way to do this is to use a random numerical value to select some entry in the table. Then we swap the just-used and selected entries (which are occasionally the same).

Operation

In operation, a data character is substituted into ciphertext or a result character. Then the substitution table is permuted. In the usual case, the content of the just-used entry is swapped with the content of some entry selected by a keyed random number stream.

Enciphering

Suppose we have a simple four-element substitution table, initialized as: **(D,B,A,C)** for addresses **A..D**. That is, address **A** contains **D**, address **B** contains **B**, and so on. In this way, the table transforms any input value in the range **A..D** into some output value in the same range. This is classical Simple Substitution.

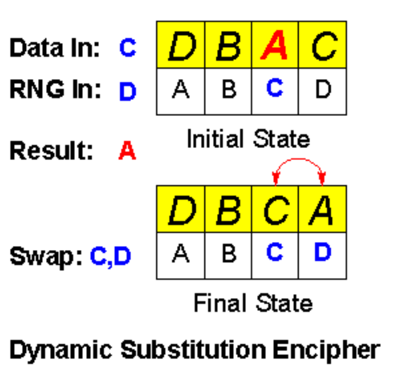
```
INITIAL STATE           Forward Table
                       --- --- --- ---
     Table Value      | D | B | A | C |
                       --- --- --- ---
     Table Location     A   B   C   D


Data In (DI):  C
Random In (RI):  D

Data Out (DO) = Fwd[DI]  =  Fwd[C]  =  A
Swap( Fwd[DI], Fwd[RI] )  =  Swap( Fwd[C], Fwd[D] )


FINAL STATE             Forward Table
                       --- --- --- ---
     Table Value      | D | B | C | A |
                       --- --- --- ---
     Table Location     A   B   C   D
```

Suppose we have a Data input of **C** and RNG input of **D**: The data value uses location **C**, which contains the value **A**; this is the result or "ciphertext." The RNG value selects location **D**, and then the values in the "used" and "selected" locations are exchanged. This produces the changed table: **(D,B,C,A)**. (This is perhaps the most efficient of many possible forms of Dynamic Substitution.)



Dynamic Substitution Encipher

In the table, each substitution element changes "at random" whenever it is used, and the more often it is used, the more often it changes. The effect of this is to adaptively "even out" symbol-frequency statistics which are the principle weakness of simple substitution. Also, the elements are changed "behind the scenes" (the exchange is not visible until future data selects one of the exchanged elements), and this tends to hide the confusion sequence.

Deciphering

Dynamic Substitution is deciphered by maintaining an inverse table, and by updating the inverse table dynamically whenever elements in the forward table are exchanged. This generally also requires maintaining a forward table.

Suppose we have the same substitution table we had before: **(D,B,A,C)**. For deciphering we also need the inverse table: **(C,B,D,A)** which we originally compute from the forward table. We assume that we will have the transmitted ciphertext and also exactly the same RNG value as at the far end. The RNG is initialized from the same key and so produces exactly the same sequence on both ends.

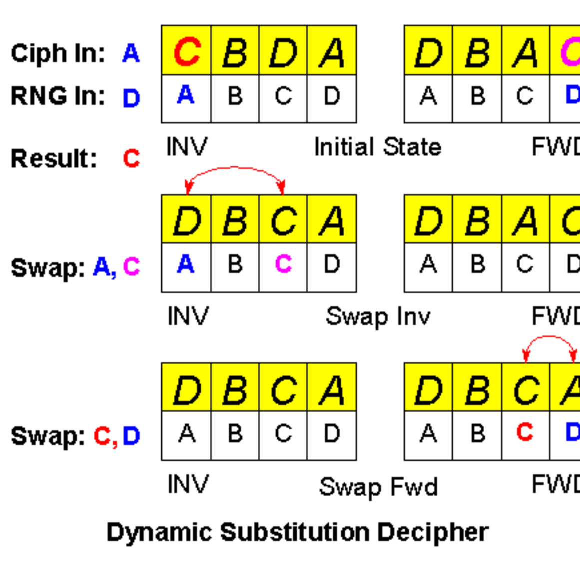
```
INITIAL STATE           Inverse Table        Forward Table
                       --- --- --- ---      --- --- --- ---
     Table Value      | C | B | D | A |    | D | B | A | C |
                       --- --- --- ---      --- --- --- ---
     Table Location     A   B   C   D        A   B   C   D


Data In (DI):  A
Random In (RI):  D

Data Out (DO) = Inv[DI]  =  Inv[A]  =  C
Swap( Inv[DI], Inv[Fwd[RI]] )  =  Swap( Inv[A], Inv[C] )
Swap( Fwd[Inv[DI]], Fwd[RI] )  =  Swap( Fwd[C], Fwd[D] )


FINAL STATE             Inverse Table        Forward Table
                       --- --- --- ---      --- --- --- ---
     Table Value      | D | B | C | A |    | D | B | C | A |
                       --- --- --- ---      --- --- --- ---
     Table Location     A   B   C   D        A   B   C   D
```

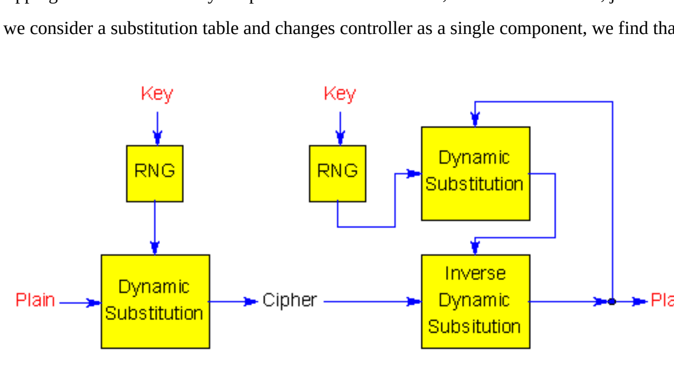
Suppose we have a Ciphertext value of **A** and RNG input of **D**: The ciphertext uses location **A** in the inverse table, which contains the value **C**, which is the result or "plaintext": our original value. The RNG selects location **D** in the forward table, which contains value **C**. Then we exchange the values in locations **A,C** (the enciphered data and enciphered RNG values) in the inverse table. Then we exchange the values in locations **C,D** (the plaintext data and plaintext RNG values) in the forward table, just as occurs during enciphering. This serves to keep both the forward and inverse tables "in sync" with the enciphering system.



Dynamic Substitution Decipher

The point of this is to minimize the amount of work we have to do to maintain the inverse table (which does the deciphering). The locations to be swapped in the inverse table are the enciphered plaintext and the enciphered RNG value. Now, the enciphered plaintext is just the ciphertext value, which we have, but the enciphered RNG value is something we do not have. Therefore, we also maintain a forward table to encipher the RNG value so we can update the inverse table. This means that the forward table also must be maintained or "kept in sync." The forward table is updated by swapping elements selected by the plaintext and RNG values, which we now have, just as occurs during enciphering.

If we consider a substitution table and changes controller as a single component, we find that one such component can be used to encipher, and two such components can be used to decipher:



The difference between the normal and inverse components is just the arrangement of the substitution table.

Design Opportunities

Dynamic Substitution can be used to replace exclusive-OR combining in stream ciphers. But when we find a component with new properties we also gain new design opportunities:

- **Use Modern VLSI Strengths:** A Dynamic Substitution table contains substantial internal state; the usual byte-wide combiner uses 256 bytes (and so has 256 *factorial* possible arrangements). This allows us to make good use of dense on-chip memory, and this is a big advantage over the limited hardware which was available to mid-70's designs. (For example, the Cloak2 design uses 17 byte-wide Dynamic Substitution combiners, each having two byte-wide tables. This is a total combiner store of 8704 bytes -- 69632 bits -- which is less than 1/2 of one percent of the store available in a single commonly-used high-production 16Mb dynamic RAM chip.)
- **Make Each Combiner Unique:** Because a Dynamic Substitution combiner contains internal state which is shuffled upon initialization, each Dynamic Substitution combiner is unique. This means that The Opponent must somehow recover or reproduce the state of each *particular* Dynamic Substitution combiner. (In contrast, exclusive-OR's *have* no state, and so provide no uncertainty which The Opponent must analyze, and thus require more strength in some other part of the system.)
- **Use Nonlinear Combiners:** Because a Dynamic Substitution combiner is shuffled, it is extremely unlikely to be linear, and continued combining operations will re-arrange it in any case. (In contrast, all additive combiners, including exclusive-OR, are absolutely linear.)
- **Use Multiple Combiners in Sequence:** Because Dynamic Substitution is nonlinear, it is reasonable to use multiple combiners in sequence. That is, by first enciphering in one combiner, and then another, we can force The Opponent to confront *both* combiners, *each* of which is separately changing dynamically and differently. (Since exclusive-OR is linear, there is very little advantage to using sequential exclusive-OR's.)
- **Use a Dynamic Selection Among Multiple Combiners:** Because multiple Dynamic Substitution combiners are likely to be very different, it is reasonable to dynamically select among an array of such combiners. For The Opponent, this means that contiguous ciphertext bytes may not have even been processed in the same combiner. (In contrast, all exclusive-OR's are exactly the same, and a selection between identical exclusive-OR's can provide no advantage at all.)

Negatives

Whenever we have a different design, we can expect that some characteristics will be disadvantageous, at least in particular situations:

- **Size:** Dynamic Substitution requires table storage; exclusive-OR does not. If we assume the usual byte-wide table, and two tables to allow deciphering, we are talking 1/2 KB per combiner. While this could be a substantial amount of store in the special case of some single-chip processors, in most cases, an extra 1/2 K is insignificant.
- **Speed:** Dynamic Substitution requires an exchange operation; exclusive-OR does not. For deciphering, we need two exchanges per byte deciphered. This implies something like 8 instructions, whereas exclusive-OR needs only one. However, in most cases, 8 simple instructions is a relatively small part of the overall computation, which typically involves reading and writing files. Normally, there is only a modest speed difference between exclusive-OR and Dynamic Substitution combining. And the added strength of Dynamic Substitution might support the use of a simpler and faster confusion generator.

  Dynamic Substitution also requires that tables be shuffled or keyed prior to use; exclusive-OR does not. Usually, though, such "initialization overhead" is a small part of the overall computation.
- **Error Expansion:** If a ciphertext transmission error occurs, Dynamic Substitution swaps the wrong element, which will affect future parts of the message; exclusive-OR does not. This is a natural consequence of the mechanism which "smoothes out" character-frequency differences in the message. Most modern communications and storage subsystems deliver virtually error-free data (they have their own error-detection and correction subsystems). Where transmission errors do occur, the data stream can be partitioned into small blocks or "lines," and any error confined to within a particular block. (See, for example, the design of the Penknife cipher.)

Strength

Dynamic Substitution is indisputably stronger than the exclusive-OR combiner used in most stream ciphers: With exclusive-OR, each byte of known-plaintext and associated ciphertext will reveal a byte from the confusion sequence. In a similar situation, Dynamic Substitution does not directly expose confusion sequence bytes.

But Dynamic Substitution is a *tool*, and is not by itself a guarantee of unbreakable strength. There are at least two methods for finding confusion sequence values from the basic unhardened design: known-plaintext and defined-plaintext.

Known Plaintext Attack

One way to deduce confusion values (in a simple system with one non-hardened combiner) is to examine the ciphertext for "close" repeats of the same value. When a ciphertext value repeats, it means that a plaintext value has found where the earlier ciphertext value was moved. This plaintext value is naively the same as the earlier confusion value.

When we use Dynamic Substitution, we expect to find particular ciphertext values separated by about the same number of steps as there are different values. But over this same period there will be about twice as many values moved in the internal table as there are values in that table. This means that repeats of the expected distance often will have moved twice, and so will *not* identify the first confusion value. For this reason, we can use only "close" repeats, and unless the repeats are adjacent, even some of *those* values will not be the desired confusion value. We might instead use only "adjacent" repeats, but these are much less frequent.

So instead of *known* confusion values, we normally get *probable* confusion values, and we find those *at random positions* in the sequence. Or, if The Opponent has enough known-plaintext and the ability to use widely-spaced values in a solution, perhaps adjacent repeats would be used. Whether or not this is a real weakness depends on various things, including the size and strength of the confusion sequence generator or random number generator (RNG):

- **Large RNG:** If The Opponent is using *probable* confusion values, and the confusion RNG is linear, with a small amount of state, it may be possible to try various sets of confusion values until the system is solved. (Although false values probably will produce *some* solution, if we run that system long enough a bad set of values should conflict with known ciphertext.) But if the RNG has a *large* amount of state, selecting a set of correct confusion values from the larger set of *probable* confusion values by trial-and-error will be very difficult.
- **Nonlinear Confusion Sequence:** Even if The Opponent has confusion sequence values, it will normally be necessary to *solve* the RNG system. One technique to complicate this is to use a nonlinear confusion sequence, which should be difficult to solve, even given the correct values. Although nonlinear confusion *generators* generally have problems, a *linear* sequence can be modified to be *nonlinear* in a number of ways. These would include discarding random amounts of the sequence at random times, and exclusive-ORing random amounts of the sequence with a random element from the sequence.
- **Multi-Level Combining:** One technique which virtually eliminates the known-plaintext weakness is to place two combinings in sequence. Although close ciphertext repetitions will still indicate the re-use of a confusion value in the second combiner, since the data value hidden between the combiners is unknown, the confusion value cannot be identified. The use of two combiners also implies that two confusion values are used when ciphering a single data value. There simply is not enough information in a single plaintext / ciphertext pair to define *two* confusion values.
- **Dynamic Combiner Selection:** Another technique which virtually eliminates the known-plaintext weakness is to dynamically select one of multiple active combiners. This makes it difficult for The Opponent to know which values come from which combiner, information which seems necessary for use of repetition data. This scheme *also* expands the amount of confusion used in ciphering, and a single plaintext / ciphertext pair simply does not have enough information to define more than a single confusion value.

As usual in cryptography, we cannot expect a simple mechanism to be secure on its own. Instead, each mechanism must be used in suitably secure ways. For Dynamic Substitution, secure techniques include sequential ciphering and dynamic combiner selection, and both of these are really only useful for nonlinear combiners with internal state.

Defined Plaintext Attack

Another way to deduce confusion (again, in a simple system with one non-hardened combiner) values is to explore the table arrangement both before and after any particular substitution occurs. That is, The Opponent tries one plaintext value, then resets the system and tries another, until all possible values have been covered. (This of course implies that The Opponent *can* reset the system, cipher any desired message, and capture the ciphertext result. The ability to reset the system is generally not available in real designs.)

If The Opponent conducts a full table exploration just *before* any particular substitution, and then again *after* that operation, typically two table elements will have changed places. This will identify the element selected by the data value and the element selected by the confusion value. But the data value is known (since The Opponent generated the message), so the confusion value can be deduced from the original state of the table.

On the other hand, "table exploration" generally implies a *defined* plaintext attack, which can be prevented in other parts of the cipher. For example, when a cipher uses a random message key, the actual data-ciphering key is a random value not under the control of the user. In general, this means that every ciphering will have a different table initialization, and this means that it will be impossible for The Opponent as a user to reset the system to a known state, even though repeatedly using the same User Key. This makes it impossible for The Opponent to conduct even *one* table exploration.

Similarly, by selecting from among an array of separate Dynamic Substitution combiners, The Opponent would seemingly need to isolate the ciphering from each individual table. And even a *small* array makes it hard to associate each result with a *particular* table, even if a defined-plaintext attack is otherwise enabled.

Hardening Dynamic Substitution

Various constructions can be used to strengthen the basic Dynamic Substitution operation, up to and including re-shuffling the entire table after every character. But the combining operation can be very significantly strengthened with far less overhead than this. Two interesting approaches may be called *Doubly Indirect Data* and *Double Random*, and these may be used either separately or together.

Doubly Indirect Data

Encipher data through the table *twice*, and exchange the second selected element with the random selected element.

Enciphering:

```
INITIAL STATE           Forward Table
                       --- --- --- ---
     Table Value      | B | D | A | C |
                       --- --- --- ---
     Table Location     A   B   C   D


Data In (DI):  C
Random In (RI):  D

Data Out (DO) = Fwd[Fwd[DI]]  =  Fwd[Fwd[C]]  =  B
Swap( Fwd[Fwd[DI]], Fwd[RI] )  =  Swap( Fwd[A], Fwd[D] )


FINAL STATE             Forward Table
                       --- --- --- ---
     Table Value      | C | D | A | B |
                       --- --- --- ---
     Table Location     A   B   C   D
```

Deciphering:

```
INITIAL STATE           Inverse Table        Forward Table
                       --- --- --- ---      --- --- --- ---
     Table Value      | C | A | D | B |    | B | D | A | C |
                       --- --- --- ---      --- --- --- ---
     Table Location     A   B   C   D        A   B   C   D


Data In (DI):  B
Random In (RI):  D

Data Out (DO) = Inv[Inv[DI]]  =  Inv[Inv[B]]  =  C
temp = Inv[DI]  =  Inv[B]  =  A
Swap( Inv[DI], Inv[Fwd[RI]] )  =  Swap( Inv[B], Inv[C] )
Swap( Fwd[temp], Fwd[RI] )  =  Swap( Fwd[A], Fwd[D] )


FINAL STATE             Inverse Table        Forward Table
                       --- --- --- ---      --- --- --- ---
     Table Value      | C | D | A | B |    | C | D | A | B |
                       --- --- --- ---      --- --- --- ---
     Table Location     A   B   C   D        A   B   C   D
```

The advantage here is the added protection of decoupling the input data from the output entry. This generally prevents The Opponent from finding a previous random value when a data in value produces a match to a preceding data out.

The cost of this added strength is quite modest, being just one more indirect access when enciphering, and perhaps a new temp variable when deciphering.

Double Random

Encipher data through the table once. To form an ultimate random value, *encipher one random input value and additively combine it with a second random input.* Then exchange the selected element with the random selected element.

Enciphering:

```
INITIAL STATE           Forward Table
                       --- --- --- ---
     Table Value      | D | B | A | C |
                       --- --- --- ---
     Table Location     A   B   C   D


Data In (DI):  C
Random In A (RIA):  D
Random In B (RIB):  C

Data Out (DO) = Fwd[DI]  =  Fwd[C]  =  A
temp = RIB + Fwd[RIA]  =  C + Fwd[D]  =  C + C  =  A
Swap( Fwd[DI], Fwd[temp] )  =  Swap( Fwd[C], Fwd[A] )


FINAL STATE             Forward Table
                       --- --- --- ---
     Table Value      | A | B | D | C |
                       --- --- --- ---
     Table Location     A   B   C   D
```

Deciphering:

```
INITIAL STATE           Inverse Table        Forward Table
                       --- --- --- ---      --- --- --- ---
     Table Value      | C | B | D | A |    | D | B | A | C |
                       --- --- --- ---      --- --- --- ---
     Table Location     A   B   C   D        A   B   C   D


Data In (DI):  A
Random In A (RIA):  D
Random In B (RIB):  C

Data Out (DO) = Inv[DI]  =  Inv[A]  =  C
temp1 = RIB + Fwd[RIA]  =  C + Fwd[D]  =  C + C  =  A
temp2 = Inv[DI]  =  Inv[A]  =  C
Swap( Inv[DI], Inv[Fwd[temp1]] )  =  Swap( Inv[A], Inv[D] )
Swap( Fwd[temp2], Fwd[temp1] )  =  Swap( Fwd[C], Fwd[A] )


FINAL STATE             Inverse Table        Forward Table
                       --- --- --- ---      --- --- --- ---
     Table Value      | A | B | D | C |    | A | B | D | C |
                       --- --- --- ---      --- --- --- ---
     Table Location     A   B   C   D        A   B   C   D
```

The advantage of this method is that we absolutely *know* that a single byte of ciphertext cannot *possibly* resolve *two* bytes of random input.

The cost, of course, is the need for twice as many random values (which may be available anyway, when using a wide Additive RNG), plus another indirect access and exclusive-OR, when enciphering. Two new temp variables may be needed for deciphering, but if that storage is available, the cost may be just the extra indirect access and exclusive-OR.

As these two hardening methods show, a wide variety of simple manipulations can be used to strengthen the fundamental combining operation. These methods can be used in addition to the ability to use a wide range of more extensive permutations far beyond just one or two swap operations. And all of these Dynamic Substitution flavors can be used in a sequence of combinings or a selection among multiple combiners.

Summary

An effective cryptographic system should be seen *as a system* designed to protect the inevitable weaknesses in each of the components. For example, the fact that exclusive-OR is trivially weak does not prevent it from being used to good advantage as a balanced mixer in many ciphering designs. But a stronger component needs less protection.

Perhaps the biggest advantages of Dynamic Substitution follow from a construction based on substantial internal state. It is the Dynamic Substitution internal state which supports the use of (nonlinear) combinings in sequence. In contrast, multiple combining rarely improves strength when using (linear) exclusive-OR. It is the Dynamic Substitution internal state which also makes each combiner different, and thus supports the use of a selection from among several combiners. And, of course, there *are* no "different" exclusive-OR combiners. The cost is a need to store and initialize this state information.

By making use of the Dynamic Substitution internal state, a cipher can remain secure *even if* weakness is later found in the confusion generator subsystem. Indeed, a simpler and faster RNG can be used from the start. These are significant advantages.

Acknowledgments

Thanks to John Savard (seward@netcom.ca) for pointing out that "close" repeats in normal Dynamic Substitution ciphertext can expose confusion values.

Thanks to Keith Lockstone (klockstone@cix.compulink.co.uk) for suggesting ways to "harden" Dynamic Substitution and prevent confusion exposure.

Terry Ritter, his current address, and his top page.

Last updated: 1997-01-16

PUBLISHED: Ritter, T. 1990. Substitution Cipher with Pseudo-Random Shuffling: The Dynamic Substitution Combiner. *Cryptologia*. 14(4): 289-303.

To read the complete article off-line, save these graphics files: Figure 1 (TEST1CGM.GIF), Figure 2 (TEST2CGM.GIF), and Figure 3 (TEST6CGM.GIF).

Substitution Cipher with Pseudo-Random Shuffling: The Dynamic Substitution Combiner

Terry Ritter

ADDRESS: Blue Jean Software, 2609 Choctaw Trail, Austin, Texas 78745.

ABSTRACT: A cipher mechanism or process which can be viewed as a modified substitution cipher. A translation table is used to replace plaintext symbols with ciphertext symbols; the modification consists of changing the contents of the translation table after each substitution. The dynamic translation table acts to confuse symbol frequency statistics and so frustrate the usual cryptanalytic attacks.

The same mechanism can also be viewed as a cryptographic combiner, and can replace the exclusive-OR combining function used in Vernam stream ciphers. The dynamic translation table acts as a one-way function to protect the pseudo- random sequence, and consequently helps to prevent cryptanalysis.

KEYWORD LIST: cryptography, computer cryptography, cipher, stream cipher, product cipher, substitution, dynamic substitution, combiner, cryptographic combiner, mixer, shuffle

---

INTRODUCTION

This paper discloses an apparently new cryptographic mechanism which can be described as dynamic substitution. Although structurally similar to simple substitution, dynamic substitution has a second data input which acts to re-arrange the contents of the substitution table. The mechanism *combines* two data sources into a complex result; under appropriate conditions, a related inverse mechanism can then *extract* one of the data sources from the result.

A dynamic substitution combiner can directly replace the exclusive-OR combiner used in Vernam stream ciphers. The various techniques used in Vernam ciphers can also be applied to dynamic substitution; any cryptographic advantage is thus due to the additional strength of the new combiner.

This paper develops a particular form of dynamic *substitution*; a related paper develops a form of dynamic *transposition*. The substitution version is generally easier to understand, more efficient, and more easily applied as a stream cipher. However, the slower transposition version may have greater cryptographic strength.

BACKGROUND

For a general background in cryptology see Kahn [15], and for details on the classical systems and their analysis see Gaines [12]. More modern statistical approaches are given by Sinkov [45] and Deavours [6]. A good partly-technical anthology is Deavours *et. al.* [5]. There is a nice but older survey by Mellen [22], a major effort by Diffie and Hellman [8], and a newer one by Massey [20] (also see the other papers in that issue). A rigorous but not always applicable theoretical base starts with Shannon [41] and is extended by Hellman [14]. Beker and Piper [1] is a good technical reference. Denning [7] and Pfleeger [32] present cryptography in the broader context of computer security issues.

THE VERNAM CIPHER

The Vernam cipher [47], although originally implemented with electromechanical relays, may well mark the start of modern cryptography. A Vernam cipher directly combines a stream of plaintext data with a pseudo-random confusion stream using what we now know of as mod 2 addition. This same combining function is also known as the Boolean logic exclusive-OR, and is widely available in digital integrated circuits and as an instruction on most computers and microcomputers.

Since each ciphertext element from a Vernam combiner is the (mod 2) sum of two unknown values, the plaintext data would seem to be well hidden. Such appearances are deceptive, however, and a Vernam cipher is susceptible to several cryptanalytic attacks, including *known-plaintext* and *probable* words [37]; if some part of the plaintext is known (or even guessed), the cryptanalyst can directly obtain some of the confusion stream [24, 25]. And if the confusion sequence can be penetrated and reproduced, the cipher is broken [34, 43, 21]. Similarly, if the same confusion sequence is ever re-used, and the overlap identified, it becomes simple to break that section of the cipher [37].

For these reasons, the modern Vernam cipher generally relies on an analysis-resistant pseudo-random sequence generator for security [e.g. 13]. But the design of such a generator is non-obvious [3], and is even more difficult than it might seem, since the cryptanalyst might well possess analytical knowledge and capabilities superior to those of the designer of the generator. Future analysts may be even more capable. Accordingly, constructs which may seem complex to the designer [13, 33, 4] may well yield, eventually, to the superior knowledge and computational resources of a cryptanalyst [43, 38, 31].

CRYPTOGRAPHIC COMBINERS

An alternate approach to the design of a secure stream cipher is to seek combining functions which can resist attack; such functions would act to hide the pseudo-random sequence from analysis [42, 43, 44]. Such *cryptographic combining functions* could be used to replace the Vernam exclusive-OR combiner (if they have an inverse) [40], or they might just combine pseudo- random sequences to make a more complex sequence [28] which is harder to analyze.

A cryptographic combiner is not intended to be an ultimate cipher; indeed, it is not at all clear that such a thing is possible. An improved combiner is intended to force cryptanalysis to be more difficult, time consuming, and expensive than would be the case with a simple combiner. The cryptographic worth of a combiner can be described as the improvement in security when compared to the now-standard Vernam exclusive-OR combiner.

The mechanism of this paper is a new combining function which extends the weak classical concept of simple substitution into a stronger form suitable for computer cryptography.

SUBSTITUTION CIPHERS

Classical simple substitution replaces each letter of the alphabet with one fixed substitute [12, 45]. Simple substitution is normally considered to be a very weak cryptographic operation [22], mainly because substitution in no way obscures the letter- frequency distribution of the source text. That is, for a particular language and topic, a statistical analysis of the enciphered data will tend to match the general statistics for that language.

The fundamental operation of substitution is pervasive in cryptography [10]. But in all known previous systems the substitution is *static*. That is, each substitution table is fully defined (either by the designer or the key) before starting encryption, and the contents of each substitution table remain unchanged for the duration of that particular cipher or message.

This paper is concerned with the cryptographic strengthening of the fundamental substitution operation through *dynamic* changes to a substitution table. The substitution table can be changed under the control of a separate data stream, usually originating from a pseudo-random sequence generator. The combination of substitution and a strategy for changing the contents of the substitution tables yields a cryptographic combining function; such a function may be used to combine plaintext data with a pseudo-random sequence to generate enciphered data. A particular strategy is presented which is applicable to computer hardware or software implementations. In many cases, such a substitution is desired to be invertible, and the presented scheme supports the efficient maintenance of an inverse.

SUBSTITUTION

In mathematical set-theoretic terms, a substitution is a relation between two sets, which is illustrated here using sets *X* and *Y*. We define a mapping function *f* from set *X* into set *Y*, and for each element *x* in set *X*, we have *y = f(x)*. In many cases, function *f* is arranged to be one-to-one (such that for each *x* in *X*, each corresponding y is unique), so that there will be an inverse function *f^-1*. If there is an inverse, for each element *x* in set *X* we have *x = f^-1(f(x))*. Thus, we can encipher with *y = f(x)* and decipher with *x = f^-1(y)*. Often, the domain (set *X*) is identically equal to the range (the subset of *Y* covered by *y = f(x)*), but this is not necessary. The symbols *xi* and *yi* will be used as a convenient notation to represent some particular data element before and after enciphering.

The simple substitution function is a monoalphabetic substitution cipher, and is easily penetrated since a mapping preserves the frequency distributions of the source material (the message). That is, for each occurrence of element *xi* from set *X*, the corresponding element *yi = f(xi)* appears in the ciphertext. Thus, whatever proportions exist among the elements *x* in the message, those same proportions will be retained among the elements *y* after substitution. It might be said that the frequency distribution characteristics are *invariant through* or *preserved by* the static substitution function.

THE SIMPLE PERMUTING MAP

But suppose the mapping function, now *f1*, is permuted to a new, slightly changed mapping function *f2* after the first element of the message (which we take to be *xi*). In particular, suppose that the mapping which takes *xi* to *yi* is transposed or exchanged with some mapping at random, say *j*. After the exchange, the map takes *xi* to *yj* and also *xj* to *yi*. Note that simply permuting a map preserves its one-to-one characteristics (assuming it originally had any), so that a different but unambiguous inverse will still exist. In this way, the just-used map element can be changed to any of the map values, and this can be done after each element of the message is enciphered. The act of exchanging an element with another selected at random is the basis for the well-known shuffle algorithm [9, 17, p. 139]; this is one of many possible strategies for changing the contents of the substitution table. The general principle of a changing substitution table can be termed *dynamic substitution*.

It seems clear that repeated application of pseudo-random map exchange operations destroys the fixed correspondence between *xi* and *yi* so that the frequency distribution invariance no longer holds. That is, since the mapping from *xi* to *yi* is shuffled whenever *xi* is enciphered, it is unlikely that the next occurrence of *xi* will also map to *yi*. This change occurs for all elements *x* in set *X* whenever an element is enciphered. Thus, this scheme would seem to obscure the frequency distribution statistics of the message. And, since the pseudo-random sequence acts only indirectly (through the exchange operation), there is some hope that it will remain hidden in any case.

THE INVERSE PERMUTING MAP

A dynamic substitution function might be used to encipher data, or it might be used to further complicate a random-number sequence. But if it is to be used to encipher data, it will be necessary to create and maintain an inverse map to support deciphering. Assuming that there are no repeated values in *f*, the initial inverse *f^-1* of *f* is easily obtained by stepping through *X*: For each value *x* in set *X*, the original map *f* generates a corresponding *y* value; that *y* value is the position in the inverse map *f^-1* where the original *x* value is placed.

Before the original map *f* is permuted, it takes *xi* to *yi* and also *xj* to *yj*; the inverse map *f^-1* will function in the reverse direction to take *yi* to *xi* and also *yj* to *xj*. After the exchange operation, the original map *f* takes *xi* to *yj* and also *xj* to *yi*; accordingly the inverse map *f^-1* must take *yi* to *xj* and also *yj* to *xi*. Thus, it is also necessary to permute the inverse map *f^-1* appropriately after each mapping operation, and this is done simply by exchanging elements *yi* and *yj* in the inverse map *f^-1*.

In practice, permuting the inverse map *f^-1* is slightly more complicated than permuting *f* alone, because the pseudo-random value (which we call *j* and which selected *xj* in *f*) must be mapped through *f* to find *yj*, which is one of the elements to exchange in *f^-1*. This scheme requires that both *f* and *f^-1* maps be present during deciphering, even if only the *f^-1* map is actually used to decipher data. But this scheme also keeps the inverse map up to date without re-generating the full inverse after every substitution operation. The other element of the "exchange pair" is *yi*, which is just the enciphered data element, and is thus already known.

DYNAMIC SUBSTITUTION

In cryptologic terms, dynamic substitution is a sort of extended substitution cipher. A substitution table is used to translate each data value into an enciphered value. But after each substitution, the table is re-ordered. At a minimum, it makes sense to exchange the just-used substitution value with some entry in the table selected at random. This generally changes the just-used substitution value to help prevent analysis, and yet retains the existence of an inverse, so that the cipher can be deciphered.

To a potential cryptanalyst, the substitution table starts out completely unknown. When a data value is translated through the table, that particular substitution is at least potentially known. But that substitution value is immediately changed, so the substitution table is again completely unknown. In this way, external cryptanalysis of the arrangement of the substitution table is greatly complicated.

Whenever a particular substitution is used, it is changed, and the more often it is used, the more often it is changed; this seems to be the *optimal* characteristic for hiding usage frequency statistics, and this is automatic and inherent in the combiner. Moreover, the pseudo-random number sequence does not directly select any data for output, it only changes the table "behind the scenes"; this gives us some grounds for asserting that the pseudo-random sequence remains hidden, to some unknown extent.

A substitution which has an inverse is normally thought to be weaker than one which does not, so it is worth noting that dynamic substitution does not have an inverse, it has a *changing* inverse, and that affects the analysis.

Dynamic substitution is one way to build a cryptographic combiner; it is not a complete cipher. However, when combined with a strong cryptographic random number generator, message keys, and other extensions, dynamic substitution can be a major part of a strong cryptographic system. It might be used simply to replace a Vernam combiner in existing equipment. It can also be used to complicate a random-number stream, or as one module in a complex multi-module ciphering system.

BLACK BOX ANALYSIS

Dynamic substitution may be considered to be a *black box*, with two input ports ("Data In" and "Random In"), and one output port ("Combiner Out"). In the simple version, each data path has similar width; evidently the mechanism inside the box in some way *combines* the two input streams to produce the output stream. It seems reasonable to analyze the output statistically, for various input streams.

Figure 1 charts the frequency distributions measured on two ports of a Simple Substitution Combiner. The "Data In" data was a sizable text file (a book chapter) with all spaces and punctuation deleted, and lower case converted to upper, leaving a 26-element alphabet of 18,135 capital letters. No random stream is used in simple substitution. Note that the "Data In" and "Combiner Out" distributions are the same, just re-arranged.

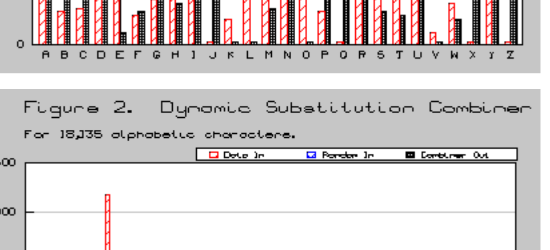


Figure 2 charts the frequency distributions measured for a Dynamic Substitution Combiner. The "Data In" stream was the same as in Figure 1. The "Random In" stream is random, with a 26-element capital letter alphabet. Note that the "Data In" and "Combiner Out" distributions are very different.

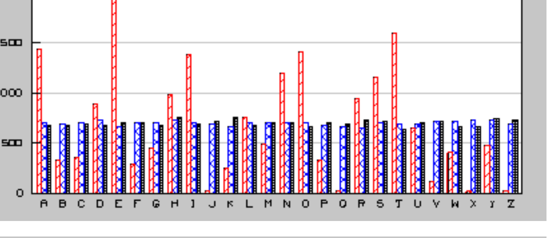
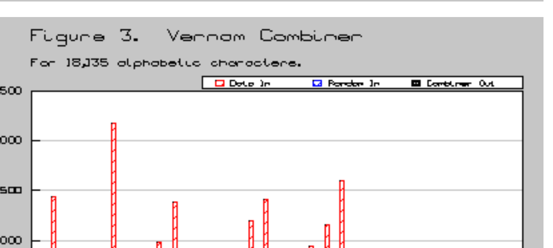


Figure 3 charts the frequency distributions for a Vernam Combiner. "Data In" was the same as before. Because the Vernam system operates on a binary representation, a Vernam "Combiner Out" stream is inherently a binary-power alphabet (in this case 32 elements); the "Random In" stream was given the same range.



Two additional experiments were conducted on Dynamic Substitution, using a constant value ('E') into one port, and a pseudo-random stream into the other port; the results were a little surprising, but their graphs provided little insight: In both cases the "Combiner Out" stream resembled a random data stream. This was unexpected because the design was intended to randomize a stream of predictable data, and not necessarily to handle a non-random "random" input, which it does. Apparently this happens because the exchange process (functioning after each substitution) takes two streams; if either one has random characteristics, the substitution table will be randomized, and this will randomize the output stream. It may be that this sort of statistical independence of the input ports is necessary for a strong cryptographic combiner. Similar results are obtained using what would normally be considered radically different inputs to the mechanism.

The last experiment was the randomization effect of a standard Vernam exclusive-OR combiner on the same constant input.

Results from these experiments were collected using common cryptographic statistical measures.

MEASURES OF RANDOMNESS

The black box test results can be summarized in the form of cryptographic *delta IC* [23], and *Z-coefficient* [6, 18] computations. In both cases, a count is made of the number of occurrences of each value in the stream being analyzed. Then a "Phi" value is computed, which is conceptually the sum of the squares of each element count. (In practice, instead of the square of the element counts n * n, the value n * (n - 1) is used [1, 45].)

The index of coincidence (IC) is conceptually *the sum of the squares* (of the element counts) *over the square of the sum* (the total element count). Because the IC depends on the size of the cryptographic alphabet, it is useful to normalize it to a delta IC by multiplying by the size of the alphabet. A delta IC value of 1.0 indicates a random distribution.

The Phi value has been computed, and an "expected" value of Phi (for a random data stream) can be derived. Similarly, a statistical variance (for a *monographic* cipher) can also be computed. The Z-coefficient is just the difference between the actual and expected Phi values, normalized by dividing by the variance. Thus, the Z-coefficient represents the extent (in statistical standard deviations) to which a particular sample differs from the expected value of a random sample. Consequently, a value of 0 would be expected, and a value between -2 and 2 would be most probable for a random sequence. The probability that a truly random sequence would produce any other Z-coefficient value can be interpreted as the area under a *bell shaped* standard normal probability curve. Table 1 summarizes the statistical results.

Table 1.

SUBSTITUTION DISTRIBUTION STATISTICS (delta IC / Z-coefficient)

	Data In	Random In	Combiner Out
The Data are 26-Letter Text			
Static Substitution	1.66 / 1684	---- / ----	1.66 / 1684
Dynamic Substitution	1.66 / 1684	1.00 / -0.9	1.00 / 1.1
Vernam (exclusive-OR)	2.04 / 2393	1.00 / -0.4	1.00 / -1.0
The Data are One Value Repeated			
Dynamic Substitution	26.0 / 35347	1.00 / -0.3	1.00 / -0.2
Same, Inputs Reversed	1.00 / -0.2	26.0 / 35347	1.00 / -0.1
Vernam (exclusive-OR)	32.0 / 39360	1.00 / 0.0	1.00 / 0.0

Apparently, dynamic substitution does *randomize* a statistically non-random input, with results similar to the standard Vernam system.

INTERNAL-KNOWLEDGE ATTACKS

A simple dynamic substitution combiner might conceivably allow some sort of insight into the pseudo-random sequence with a known-plaintext attack. For example, after *xi* is enciphered to *yi*, an exchange occurs as a result of a particular pseudo-random value (*j*). If another *xi* occurs immediately, the resulting *yj* mapped value might somehow provide some insight into the value (*j*) taken from the pseudo-random sequence. Note that the random sequence value *j* is not directly available in this way, but only the mapped value. And if *xi* does not recur immediately, the mapping it eventually reveals may be the result of multiple exchanges, and so would be much less useful.

Another approach would be to concentrate on the next occurrence of *yi* in the ciphertext, which, if it recurred immediately, might somehow provide a similar insight to the preceding swap value (*j*). Since both a subsequent *xi* and a subsequent *yi* might somehow provide insight to the same swap value, it may be possible to achieve a likely confirmation. Again, this would be the *mapped* value (*xj*), and not directly the desired pseudo-random value (*j*) itself. It is not clear how this information could be used to penetrate the cipher.

Penetration would seem to be easier if the known-plaintext *x* values were close together, because this would make a re-mapped mapping less likely. Thus, there is some reason to suspect that an uneven plaintext distribution (which is normal) would have some effect in making the system somewhat more vulnerable. But a good value distribution could be enforced by first randomizing the data with a Vernam (exclusive-OR) combiner (and a random-number stream) prior to dynamic substitution, or by using an additional level of dynamic substitution.

In any case, it is the responsibility of the rest of the system to assure that the same pseudo-random sequence will "never" be re-used. This is a normal requirement for Vernam stream ciphers, and is often handled with a *message key* [1].

POLYALPHABETIC DYNAMIC SUBSTITUTION

An obvious countermeasure to known-plaintext and chosen-plaintext attacks would be to use multiple different dynamic substitution maps (a polyalphabetic dynamic substitution cipher) [e.g. 1], and to select between them using a hidden pseudo-random sequence. Since consecutive xi elements would generally be enciphered in different maps, the use of repeated xi elements leads to the probability that some maps will be entered multiple times (on the same mapping) before all of the maps have been entered once, and this seems to substantially complicate cryptanalysis.

Moreover, the normal attacks on a polyalphabetic cipher are also statistical, and these seem likely to be complicated by the anti-statistical properties of the underlying permuting maps. Because the multiple maps are to be used at pseudo-random instead of in rotation, it would seem to be difficult for an analyst to isolate any particular map on which to work.

REAL CIPHER SYSTEMS

In broad terms, a Vernam stream cipher consists of an exclusive-OR combiner and some sort of confusion (random number) generator. However, real systems must be considerably more complex than this, since any cipher is only as strong as its weakest link.

Even a strong combiner and confusion generator can be made irrelevant if the system supports only a small number of keys, since simply trying all the keys (a *key search* attack) would be sufficient to penetrate such a cipher. Any real system based on keys must support enough keys to prevent this attack, but this is really more of an issue for the confusion generator than for the combiner.

There is a large body of literature on various sorts of pseudo-random number generators and some amount of work on their abilities to resist cryptanalysis. However, this literature is equally applicable to systems containing either dynamic substitution or Vernam exclusive-OR combiners, and so is essentially irrelevant to comparisons between the two.

INTERNAL STATE

Dynamic substitution combiners inherently contain internal *state* data (in the finite automata sense [e.g. 8, p. 415]), while the exclusive-OR does not. This internal state data must be initialized before ciphering, and is continuously re-ordered as a consequence of both incoming data streams; thus, the internal state is a function of initialization and all subsequent data and confusion values. Consequently, if data errors occur during communication of the ciphertext, the deciphering process will deviate from the expected sequence, and will not re-synchronize. This problem is mitigated by the widespread use of the error-detection codes (e.g., CRC) and the error-correcting block re-transmissions now commonly used in data communications. Although there can be no such thing as an error rate of absolutely *zero*, computer data can be stored or communicated with *arbitrarily* low error rates based upon the relative economic impacts of errors versus error-protection. In practical terms, even very simple "binary protocols" (e.g., XMODEM) only rarely allow any errors to survive uncorrected, so error sensitivity is not nearly the problem it was in the days of manual telegraph operators or mechanical typing-telegraph machines.

The changing internal state of dynamic substitution is the source of its strength, and that state is affected by both input sequences. Thus, it should come as no surprise that the new combiner is not as tolerant of data errors as its weaker cousin, the exclusive-OR.

APPLICATIONS

Any substitution is a mapping (from input to output), and a mapping is the most general function possible; thus, it may be used to replace a network of simple logic operations. In particular, dynamic substitution might be designed to replace substitution-permutation functions [10, 11, 16] (although large S-P functions might imply an exceedingly large map). In some cases, dynamic permutations of existing S-P maps may be used to strengthen those designs.

One use for dynamic substitution would be as a combining or mixing function on data streams. For example, dynamic substitution might easily replace the exclusive-OR logic function in a Vernam cipher. Or dynamic substitution could be used to combine two pseudo-random sequences [27, 28], in which case an inverse would be unnecessary.

Alternately, by making the domain and range of the substitution maps (f and f^-1) the same, dynamic substitution can be used in a product cipher [41], as one element in a chain or network of ciphering functions or modules.

CONCLUSIONS

The simple substitution cipher--normally one of the weakest in cryptography--becomes substantially stronger when the substitution table is changed during operation. By re-mapping at least the just-used symbol, the cipher acts to hide letter-frequency statistics, which have previously been the main avenue of entry into such a cipher. The polyalphabetic version seems even stronger.

ACKNOWLEDGMENTS

Many thanks to the referees for pointing out ambiguous discussions and important unmentioned issues. This work has improved because of their efforts.

REFERENCES

1. Beker, H. and F. Piper. 1982. *Cipher Systems.* New York: John Wiley & Sons.

2. Beker, H. and F. Piper. 1985. *Secure Speech Communications.* London/Orlando: Academic Press.

3. Blum, L., M. Blum, and M. Shub. 1983. Comparison of Two Pseudo-Random Number Generators. *Advances in Cryptology--Proceedings of Crypto 82.* New York: Plenum Press. 61-78.

4. Ciarcia, S. 1986. Build a Hardware Data Encryptor. *Byte.* September. 97-111.

5. Deavours, C., D. Kahn, L. Kruh, G. Mellen, and B. Winkle. 1987. *Cryptology Yesterday, Today, and Tomorrow.* Norwood, Mass: Artech House.

6. Deavours, C. 1987. *Cryptanalytic Programs for the IBM PC.* Laguna Hills, CA: Aegean Park Press.

7. Denning, D. 1982. *Cryptography and Data Security.* Reading, Mass: Addison-Wesley.

8. Diffie, W. and M. Hellman. 1979. Privacy and Authentication: An Introduction to Cryptography. *Proceedings of the IEEE.* 67: 397-427.

9. Durstenfeld, R. 1964. Algorithm 235, Random Permutation, Procedure SHUFFLE. *Communications of the ACM.* 7:420

10. Feistel, H. 1973. Cryptography and Computer Privacy. *Scientific American.* 228: 15-23.

11. Feistel, H., W. Notz, and J. L. Smith. 1975. Some Cryptographic Techniques for Machine-to-Machine Data Communications. *Proceedings of the IEEE.* 63: 1545-1554.

12. Gaines, H. 1956 (original publication 1939). *Cryptanalysis.* New York: Dover Publications.

13. Geffe, P. 1973. How to protect data with ciphers that are really hard to break. *Electronics.* January 4. 99-101.

14. Hellman, M. 1977. An Extension of the Shannon Theory Approach to Cryptography. *IEEE Transactions on Information Theory.* IT23: 289-294.

15. Kahn, D. 1967. *The Codebreakers.* New York: Macmillan.

16. Kam, J. and G. Davida. 1979. Structured Design of Substitution-Permutation Encryption Networks. *IEEE Transactions on Computers.* 28: 747-753.

17. Knuth, D. 1981. *The Art of Computer Programming,* Vol. 2 *Seminumerical Algorithms.* 2nd ed. Reading, Mass: Addison-Wesley.

18. Kullback, S. 1976 (original publication 1938). *Statistical Methods in Cryptanalysis.* Laguna Hills, CA: Aegean Park Press.

19. MacLaren, M. D. and G. Marsaglia. 1965. Uniform Random Number Generators. *Journal of the ACM.* 12: 83-89.

20. Massey, J. 1988. An Introduction to Contemporary Cryptology. *Proceedings of the IEEE.* 76: 533-549.

21. Meier, W. and O. Staffelbach. 1988. Fast Correlation Attacks on Stream Ciphers (extended abstract). *Advances in Cryptology--Eurocrypt 88.* New York: Springer-Verlag. 301-314.

22. Mellen, G. 1973. Cryptology, Computers, and Common Sense. *National Computer Conference, 1973, Proceedings.* 569-579.

23. Mellen, G. 1983. Cryptanalysts' Corner. *Cryptologia.* 7: 371.

24. Meyer, C. and W. Touchman. 1972. Pseudorandom codes can be cracked. *Electronic Design.* 23: 74-76.

25. Meyer, C. 1973. Design considerations for cryptography. *National Computer Conference, 1973, Proceedings.* 603-606.

26. Meyer, C. and S. Matyas. 1982. *Cryptography: A New Dimension in Computer Data Security.* New York: John Wiley & Sons.

27. Michener, J. 1985. The "Generalized Rotor" Cryptographic Operator and Some of Its Applications. *Cryptologia.* 9: 97-113.

28. Michener, J. 1987. The Use of Complete, Nonlinear, Block Codes for Nonlinear, Noninvertible Mixing of Pseudorandom Sequences. *Cryptologia.* 11: 108-111.

29. Michener, J. 1987. The Application of Key Dependent and Variable Rotor Sets to Generalized Rotor Cryptographic Systems. *Cryptologia.* 11: 166-171.

30. Michener, J. 1988. A Tool for Secret Key Cryptography. *Dr. Dobb's Journal.* August. 50-55, 96.

31. Pearson, P. 1988. Cryptanalysis of the Ciarcia Circuit Cellar Data Encryptor. *Cryptologia.* 12: 1-9.

32. Pfleeger, C. 1989. *Security in Computing.* Englewood Cliffs, New Jersey: Prentice Hall.

33. Pless, V. 1977. Encryption Schemes for Computer Confidentiality. *IEEE Transactions on Computers.* C26: 1133-1136.

34. Reeds, J. 1977. "Cracking" a Random Number Generator. *Cryptologia 1.* (also [5, pp. 509-515]).

35. Retter, C. 1984. Cryptanalysis of a MacLaren-Marsaglia System. *Cryptologia.* 8: 97-108. (Also see letters and responses: *Cryptologia.* 8: 374-378).

36. Retter, C. 1985. A Key Search Attack on MacLaren-Marsaglia Systems. *Cryptologia.* 9: 114-130.

37. Rubin, F. 1978. Computer Methods for Decrypting Random Stream Ciphers. *Cryptologia 2.* (also [5, pp. 493-508]).

38. Rubin, F. 1979. Decrypting a Stream Cipher Based on J-K Flip-Flops. *IEEE Transactions on Computers.* C28: 483-487. (also *Cryptologia* Vol. 5, and [5, pp. 283-293]).

39. Rubin, F. 1987. Foiling an Exhaustive Key-Search Attack. *Cryptologia.* 11: 102-107.

40. Sancho, J. 1987. Enumeration of Multivariable Decipherable Boolean Functions. *Cryptologia.* 11: 172-180.

41. Shannon, C. 1949. Communication Theory of Secrecy Systems. *Bell System Technical Journal.* 28: 656-715.

42. Siegenthaler, T. 1984. Correlation-Immunity of Nonlinear Combining Functions for Cryptographic Applications. *IEEE Transactions on Information Theory.* IT30: 776-780.

43. Siegenthaler, T. 1985. Decrypting a Class of Stream Ciphers Using Ciphertext Only. *IEEE Transactions on Computers.* C34: 81-85.

44. Siegenthaler, T. 1986. Design of Combiners to Prevent Divide and Conquer Attacks. *Advances in Cryptology--CRYPTO '85, Proceedings.* New York: Springer-Verlag. 273-279.

45. Sinkov, A. 1966. *Elementary Cryptanalysis: A Mathematical Approach.* Washington, DC: The Mathematical Association of America.

46. Stahl, F. 1973. A homophonic cipher for computational cryptography. *National Computer Conference, 1973, Proceedings.* 565-568.

47. Vernam, G. 1926. Cipher Printing Telegraph Systems. *Transactions AIEE.* 45: 295-301.

BIOGRAPHICAL SKETCH

Terry Ritter is a registered Professional Engineer, a member of IEEE and ACM, with a background in computer architecture, hardware, software, and now, library research. He has enjoyed spending the past few years being Blue Jean Software and Blue Jean Computer Engineering.

Terry Ritter, his current address, and his top page.

Last updated: 1996-01-04

United States Patent 5,623,549

Cipher Mechanisms with Fencing and Balanced Block Mixing

Terry Ritter

For informational purposes only. Taken from the original Application and Amendment files which were sent to the PTO in printed form. At the PTO, the documents were manually transcribed into the printed patent. Here, the files were converted to HTML, partly by hand. For these reasons this version may differ, somewhat, from the printed patent. If there is any question, see an actual printed patent copy from the PTO, and any Certificate of Correction.

The extensive use of inline images, which make the text easier to follow, caused extensive disk swapping in 16MB machines. Consequently, the document is broken into parts A, B, C and D.

Contents

- Title
- References
- Abstract
- Drawings
- Body
  - Background of Invention
    - Field of Invention
    - Description of Related Art
  - Summary of Invention
  - Brief Description of the Drawings
  - Description of the Presently Preferred Exemplary Embodiments
    - Increasing the Block Size of a Block Cipher using Balanced Block Mixing
    - Balanced Block Mixers
    - Creating and Generating Balanced Block Mixers
    - Construction of Orthogonal Latin Squares
    - Notes on Polynomial Arithmetic
    - Notes on Fields
    - Examples of Balanced Block Mixers
    - Fenced Cryptographic Mechanisms
    - 1x Fenced Cipher Mechanisms
    - Substitution Mechanisms
    - 1x Fenced DES
    - Generating Keys for the Substitution Mechanisms
    - 2x Fenced Cipher Mechanisms
    - 2x Fenced DES
    - 4x Fenced Cipher Mechanisms
    - 4x Fenced DES
    - Some Other Fenced Block Cipher Mechanisms
    - Balanced Block Mixers in Combination with Substitution Mechanisms
    - A Multiple Size Cryptographic Mechanism.
- Claims

United States Patent	**5,623,549**
Ritter	**Apr. 22, 1997**

Cipher mechanisms with fencing and balanced block mixing

Inventors: **Ritter; Terry F.** (2609 Choctaw Trail, Austin, TX 78745).

Appl. No.: **380,960**
Filed: **Jan. 30, 1995**

Intl. Cl. :	**H04L 9/18**
U.S. Cl.:	**380/37**; 380/42
Field of Search:	380/37, 42, 36

References Cited

U.S. Patent Documents

4,157,454	Jun., 1979	Becker	380/37
4,168,396	Sept., 1979	Best	380/37
4,195,200	Mar., 1980	Feistel	380/37
4,316,055	Feb., 1982	Feistel	380/37
5,003,596	Mar., 1991	Wood	380/37
5,003,597	Mar., 1991	Merkle	380/37

Foreign Patent Documents

58-201436	Nov., 1983	JP	380/37
5-95350	Apr., 1993	JP	380/37

Other References

"Keyed Balanced Size-Preserving Block Mixing Transforms," Ritter (Mar. 12, 1994) at Internet Newsgroup sci.crypt.publication.

Schweier, (1996) "Applied Cryptography" 2nd Ed John Wiley & Sons p. 50.

"Network Security", Section 2.6 Hash Algorithms, Kaufman et al (1995) pp. 53 & 486.

"Parallel FFT Hashing", Schnorr et al. pp. 1-2 delivered at Fast Software Encryption (Dec. 1993) Cambridge, England.

"Black Box Cryptanalysis of Hash Networks Based on Multiplications," Schnorr et al (May 9-12, 1994) presented at Eurocrypt 1994, Perugia, Italy.

The Efficient Generation of Cryptographic Confusion Sequences, Terry Ritter, Cryptologia, 15(2): 81-139, 1991.

Ritter/Biham Electronic Mail Conversations, dated Feb., 1994.

Keyed Balanced Size-Preserving Block Mixing Transforms, by Terry Ritter, dated Mar. 12, 1994.

Fenced DES, by Terry Ritter, dated Apr. 17, 1994.

The context of the Fenced DES Design, by Terry Ritter, dated Jun. 30, 1994.

Ritter, "Strong Block Ciphers from Weak Ones: NxM DES A New Class of DES Operating Modes", Ritter Software Engineering, Posted on Internet Jan. 31, 1994.

Ritter, "Substitution Cipher With Pseudo-Random Shuffling: The Dynamic Substitution Combiner", Cryptologia, Oct. 1990, vol. XIV, No. 4, pp. 289-304.

H. Kull and E. Specker, "Direct Construction of Mutually Orthogonal Latin Squares", Computational Theory and Logic, 1987, pp. 224-236.

Portz, "On the Use of Interconnection Networks in Cryptography", Advances in Cryptology--Eurocrypt 1991, pp. 302-315.

Kam et al., "Structured Design of Substitution-Permutation Encryption Networks", IEEE Transactions On Computers, Oct. 1979, vol. C-28, No. 10, pp. 747-753.

Feistel, "Cryptography and Computer Privacy", Scientific American, May 1973, vol. 228, No. 5, pp. 15-23.

Massey, "Safer K-64: A Byte-Oriented Block-Ciphering Algorithm", Fast Software Encryption, 1994, Ross Anderson ed. Springer-Verlay, pp. 1-17.

Lai et al., "A Proposal for New Block Encryption Standard", Advances in Cryptology--Eyrocrypt '90, 1990 pp. 389-404.

Massey et al., "Non-Expanding, Key-Minimal, Robustly-Perfect, Linear and Bilinear Ciphers", Advances In Cryptology--Eurocrypt '87, pp. 237-247.

Stevens, "The Completely Orthogonalized Latin Square", Annals of Eugenics, 1939, 9:82-93.

"Parallel FFT-Hashing" C.P. Schnorr & S. Vaudenay Fast Software Encryption, Cambridge Security Workshop Cambridge, U.K., Dec. 9-11, 1993, Proceedings Springer-Verlag.

"Black Box Cryptanalysis of Hash Networks Based on Multipermutations" Schnorr & Vaudenay Advances in Cryptology--Eurocrypt '94 Springer-Verlag.

Primary Examiner: Cangialosi; Salvatore

ABSTRACT

An enhanced cryptographic mechanism employs Latin square derived balanced size-preserving block mixers and strong, practical fencing arrays of substitution mechanisms in combination with each other and with block ciphers. Ciphers are expanded into efficient, larger, stronger versions. Block ciphers, in combination with balanced block mixers and/or with substitution mechanisms, produce cryptographic mechanisms with block sizes that are combinations of the sizes of the block ciphers. Ciphers using large data blocks can reduce data expansion to levels normally consistent with small blocks. Different sized enhanced cryptographic mechanisms are used in a multiple-size cryptographic mechanism to minimize wasted block space in a ciphered message. The cryptographic mechanism provides at least three layers of processing. In one embodiment a message passes through a fencing array of substitution mechanisms, balanced block mixers, multiple block ciphers, balanced block mixers, and another fencing array of substitution mechanisms, for encryption and decryption, yet still ciphers at a rate near that of the block ciphers alone.

10 Claims, 22 Drawing Figures

DRAWINGS

FIGURE 1: A Cryptographic Mechanism.

FIGURE 2: An Implementation of a Cryptographic Mechanism.

FIGURES 3(a) and 3(b): A Communication System Employing a Cryptographic Mechanism; and A Schematic Diagram of that Communication System.

FIGURE 3(c): A Computer System Employing a Cryptographic Mechanism.

FIGURE 4(a): Mixing - Enciphering - Mixing with a Double-Width Block.

FIGURE 4(b): Mixing - Deciphering - Mixing with a Double-Width Block.

FIGURE 5(a): Enciphering - Mixing - Enciphering with a Double-Width Block.

FIGURE 5(b): Deciphering - Mixing - Deciphering with a Double-Width Block.

FIGURE 6: Feistel Ciphering - Mixing - Feistel Ciphering with a Double-Width Block.

FIGURE 7: Mixing - Ciphering - Mixing with a Quad-Width Block.

FIGURE 8: Ciphering - Mixing - Ciphering, with a Quad-Width Block.

FIGURES 9(a) and 9(b): Balanced Block Mixers.

FIGURE 10: 1x Fenced DES; Substitute - Cipher - Substitute Cryptographic Mechanism.

FIGURE 11: A Substitution Mechanism.

FIGURE 12: Substitute - Cipher - Substitute with a 128-bit Block.

FIGURE 13: 2x Fenced DES; Substitute - Mix - Cipher - Mix - Substitute with a Double-Width Block.

FIGURE 14: 4x Fenced DES; Substitute - Mix - Cipher - Mix - Substitute with a Quad-Width Block; Power-of-2 Mixing.

FIGURE 15(a): Substitute - Cipher - Substitute.

FIGURE 15(b): 4x Fenced DES; Substitute - Mix - Cipher - Mix - Substitute with a Quad-Width Block; Power-of-2 Mixing, Reversed Order.

FIGURE 16: Cipher Formed from Small Substitutions: Substitute - Mix - Substitute - Mix - Substitute; Power-of-2 Mixing.

FIGURE 17: Cipher Formed from Small Substitutions; Substitute - Mix - Substitute - Mix - Substitute; FFT-style Mixing.

FIGURE 18: A Multiple Block-Size Cryptographic Mechanism.

BODY

CIPHER MECHANISMS WITH FENCING AND BALANCED BLOCK MIXING

BACKGROUND OF THE INVENTION

1. Field of the Invention

This invention relates to cryptography, and more particularly, to constructing and enhancing cryptographic mechanisms.

2. Description of Related Art

Cryptographic mechanisms are used in many fields to improve the security of stored and transmitted data and signals, to authenticate data and signals and to provide protection and privacy of data and signals. Cryptographic mechanisms are used, for example, in communication systems such as digital telephone systems (for both audio and video communications) and in data processing systems to maintain privacy and secrecy of stored and transmitted data, and to authenticate signals and data.

The transformation of signals or data so that their contents cannot be determined is called encryption. The re-transformation of an encrypted signal or data to get back the original signal or data is called decryption. The combination of encryption and decryption is called cryptography and is carried out by cryptographic mechanisms. For example, **FIGURE 1** shows a typical cryptographic mechanism **100**. Cryptographic mechanism **100** includes an encryption mechanism component **102** and a decryption mechanism component **104**. Data encryption operates via the encryption mechanism **102** on an input signal or data (plaintext) P (via a channel **106**) to produce an encrypted output signal or data (ciphertext) C (via a channel **108**). Similarly, the ciphertext C is decrypted using the decryption mechanism **104** to produce the plaintext P. Most cryptographic mechanisms **100** use cryptographic keys **110**, **112** to select the transformation used in encryption and decryption. A cryptographic mechanism **100** can viewed as a filter, taking a first signal and producing another signal corresponding to the encryption or decryption (as appropriate) of the first signal.

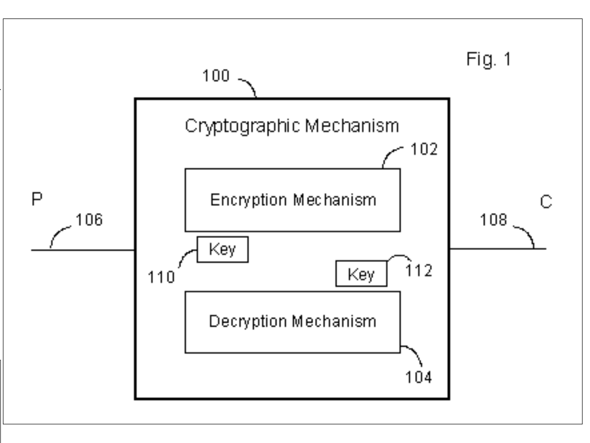


Fig. 1

Cryptographic mechanisms can be broadly categorized into two classes, namely stream and block devices. In stream cryptographic mechanisms, the input plaintext is encrypted one unit (e.g., one bit or character) at time. Decryption in these stream devices also takes place one unit at a time. In block cryptographic mechanisms, the plaintext is encrypted (and decrypted) in blocks of units, e.g., 64-bit blocks.

Cryptographic mechanism **100** may be implemented entirely in electronic hardware, entirely in software operating on a computer, or in a hybrid combination of both hardware and software. **FIGURE 2** is a schematic diagram of a hybrid implementation of cryptographic mechanism **100** which employs one or more cipher mechanisms **114**, one or more CPUs **116**, and memory **118**. Each of the encryption mechanism **102** and the decryption mechanism **104** may have its own cipher mechanism **114**, CPU **116**, and memory **118**, or some or all of these components may be shared. Cipher mechanism **114** implements a particular cipher and may be on its own chip and include its own computer. Other aspects of the cryptographic mechanism **100** are implemented using the CPU **116** and the memory **118**, in combination with the cipher mechanism **114**.

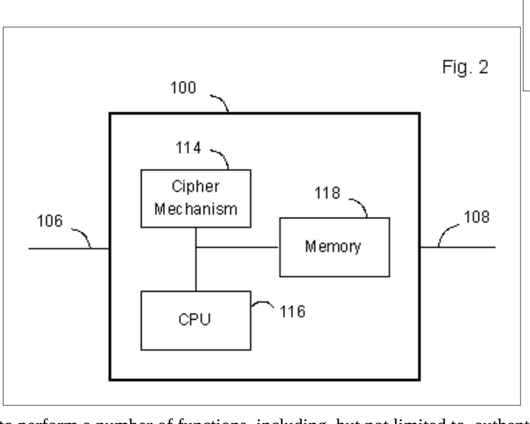
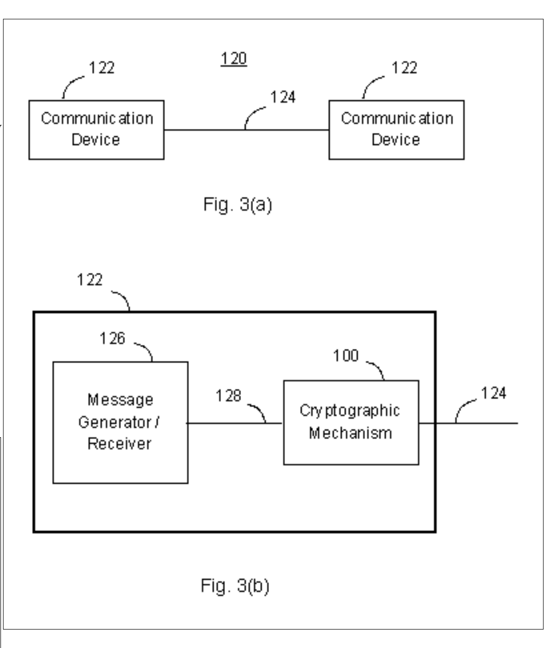


Fig. 2

Cryptographic mechanisms are used in many systems to perform a number of functions, including, but not limited to, authenticating and verifying data (transmitted or stored), hashing data, and encrypting and decrypting messages, data and signals. As examples, a communications system and a computer system, both employing cryptographic mechanisms for various purposes, are described herein. These systems are described only as examples of systems which employ cryptographic mechanisms and of the uses of cryptographic mechanisms in those systems. These examples are not intended to be a complete enumeration of all possible systems in which a cryptographic mechanism of the type described herein could be used, nor are they intended to be a complete enumeration of the various uses of cryptographic mechanisms in those systems.

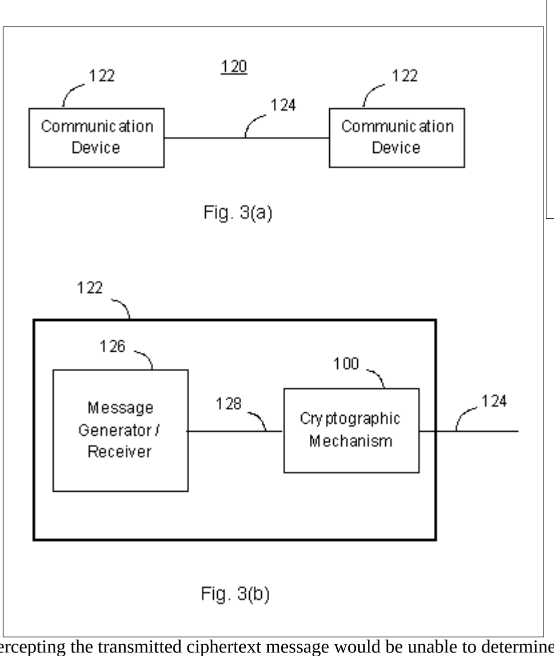
Cryptographic mechanisms are used in a typical communication system **120**, such as shown in **FIGURE 3(a)** to ensure, inter alia, the secrecy and privacy of communications within the system. In communication system **120**, communication devices **122** communicate messages over channels **124**. Communication devices **122** may, for example, be computer systems, telephones, facsimile devices, or the like, alone or in combination. The channels **124** may, for example, be regular telephone wires, fiber optic cables, radio waves, microwaves, or the like, alone or in combination. The messages transmitted over the channels **124** may be voice or data, they may be video signals, pictures, or any other form of transmittable data.



Many data communication systems are open, having some or all of the channels **124** insecure. This is especially the case when one or more of the channels employs radio waves which are detectable and easily monitored without affecting the communication system itself. For example, in the case where the communication system **120** is a telephone network, with some of the devices **122** being cellular telephones, some of the channels **124** being radio channels, and the messages being telephone conversations, i.e., voice messages, between users of telephones, the contents of the messages which are sent over radio channels **124** can be monitored by listening in on the appropriate broadcast frequency.

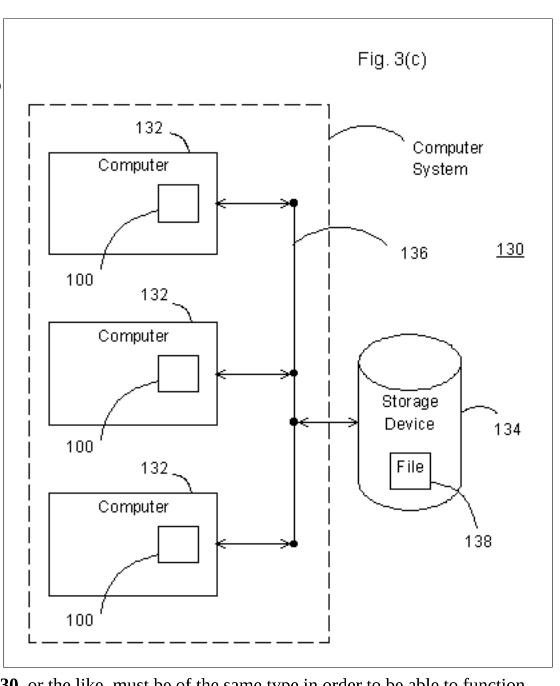
To overcome problems presented by open, insecure channels in communication systems, it is desirable to make it difficult, if not impossible, for someone to ascertain the content of a monitored message. To this end, it is desirable to transform or encode messages, prior to their transmittal over open channels, into messages whose contents cannot be determined. Clearly it is necessary to be able to transform an encoded, transmitted message back to its original message so that the intended recipient can determine its contents.

Thus, in order to overcome the insecurity of communication messages between communication devices **122** over open channels **124** in communication system **120**, it is desirable to encrypt messages prior to their transmittal by a communication device **122** and to decrypt them upon receipt by another communication device **122**. In this way, by the time a message can be intercepted, i.e., by the time a message is available on an open channel, its contents cannot be determined without being decrypted. To this end, with reference to **FIGURES 3(a)** and **3(b)**, each communication device **122** has a message generator/receiver **126** connected via internal (secure) channel **128** to cryptographic mechanism **100**. (Cryptographic mechanism **100** corresponds to the cryptographic mechanism **100** of **FIGURE 1**.) As noted above, a typical cryptographic mechanism **100** is keyed, that is, it uses a cryptographic key **110** or **112** to control its encryption/decryption functions. Communication between communication devices **122** over open channels **124** takes place via cryptographic mechanisms **100**. For example, in a transmit mode, the message generator/receiver **126** of one communication device **122** produces a plaintext message which is sent over internal channel **128** to cryptographic mechanism **120** of that communication device **122**. Cryptographic mechanism **100** uses a cryptographic key **110** to control the encryption of the plaintext message, producing a ciphertext message which is transmitted over open channel **124**. Anyone intercepting the transmitted ciphertext message would be unable to determine its contents (i.e., the plaintext message) without first decrypting the message. The ciphertext message is sent, via channel **124** to another communication device **122**. At the other communication device **122**, cryptographic mechanism **100** receives the ciphertext message and decrypts it using the same cryptographic key **112**, producing the plaintext message. The plaintext message is then sent over the internal channel **128** of communication mechanism **122** to the message generator/receiver **126**.



The internal channel **128** may be a real channel in a device or it may be a logical (i.e., virtual channel). In other words, the cryptographic mechanism **100** may be integrated with the message generator/receiver **126** in such a way that the actual channel **128** is not physically discernable or it may be a separate device connected via an internal data bus of some sort.

Cryptographic mechanisms are used in computer systems such as computer system **130** shown in **FIGURE 3(c)**, having one or more computers **132** connected to each other and to at least one storage device **134** via a bus **136**. Storage device **134** may be fixed, e.g., a hard disk, or it may be portable, e.g., a floppy disk, an optical disk, a tape, or the like. Each computer **132** may also have its own local storage device (not shown). In computer system **130**, a cryptographic mechanism **100** (**FIGURE 1**) is used, inter alia, to verify data (stored on a computer **132** or a storage device **134**, or transmitted between computers **132**), to keep data secret and private, and to prevent unauthorized access to data (stored on a storage device **134** or a computer **132**, or transmitted between computers **132**).



For example, with reference to **FIGURE 3(c)**, a computer **132** in computer system **130** stores (via bus **136**) data in a file **138** on a storage device **134**. In order to prevent unauthorized access to the data in the file **138**, whether by other users of the computer system **130**, or by someone gaining access to the file **138** in some other way, it is desirable to store the data in an encrypted form. Note that access to the data can be gained by obtaining the storage device **134** or by gaining access to the bus **136** while the data is being transferred from the computer **132** to the storage device **134**. To this end, the computer **132** includes a cryptographic mechanism **100** (**FIGURE 1**) for encrypting and decrypting stored data.

As a further example of the use of a cryptographic mechanism, in the communication system **120** of **FIGURE 3(a)**, or computer system **130** of **FIGURE 3(c)**, or the like, it is sometimes desirable to use only one aspect of the cryptographic mechanism **100**. In particular, it is sometimes desirable to use only the encryption mechanism (reference numeral **102** of **FIGURE 1**) to encrypt data in order to authenticate the data. To this end, the cryptographic mechanism **100** can be used to provide an authentification function. Data or signals to be authenticated are encrypted or hashed (controlled by a cryptographic key) by the cryptographic mechanism **100** to produce a value representative of the data. If the data changes then its corresponding encryption value will change, thereby allowing authentification.

In the cryptographic mechanisms described herein, the manner in which keys are provided to the mechanisms is well known and therefore is not described in detail. For example, each communication device **122** or computer **132** may have some means for creating, storing and inputting keys when need, and for providing these keys to their cryptographic mechanisms in a conventional manner.

The above examples (the communications system and the computer system) are only some examples of fields, mechanisms, devices and systems which employ cryptographic mechanisms, and they are not intended to be a complete enumeration of all possible systems in which a cryptographic mechanism of the type described herein could be used.

Cryptographic mechanisms **100** employed in the same system, whether it be a communication system **120**, a computer system **130**, or the like, must be of the same type in order to be able to function together. This is not to say that the mechanisms must use the same implementations, only that the mechanisms must perform the same encryption and decryption functions. For example, cryptographic mechanism **100** may be implemented entirely in electronic hardware, entirely in software operating on a computer, or in a hybrid combination of hardware and software (as shown above with respect to **FIGURE 2**). Such differently implemented mechanisms can readily be employed in the same system as long as the mechanisms perform the same or compatible encryption and decryption functions.

Throughout this application, including the claims, the term "mechanism" is intended to include all possible implementations (such as electronic hardware, computer implemented software, hybrids, or the like) or the process performed. Thus a cryptographic or cipher mechanism refers to either a process of enciphering and/or deciphering or any implementation (for example, hardware, computer implemented software, hybrid, or the like) for performing the process. Furthermore, the terms "cipher" and "ciphering" are employed to refer generally to both enciphering and deciphering.

To ensure that various cryptographic mechanisms perform the same encryption and decryption functions, various national and international standards have been set for cryptographic mechanisms and techniques. One example is the Data Encryption Standard (DES). DES is a block cipher that encrypts data in 64-bit blocks. DES takes a 64-bit plaintext and a key, effectively 56 bits long, to produce a 64-bit ciphertext. Decryption in DES takes 64-bit blocks of ciphertext and the same key to reproduce 64-bit blocks of plaintext. The effective DES key length is 56 bits (usually the key is expressed as a 64-bit number, with every eighth bit used for parity checking).

DES has been adopted as an American National Standards Institute (ANSI) standard (ANSI X3.92) and, as part of the standard, the National Institute of Standards and Technology (NIST) validates hardware and firmware implementations of DES. This validation confirms that the implementation follows the standard. The standard does not allow software versions of DES, unless they are unchangeably fixed within a hardware device (for example, by using a read- only memory to hold the hardware control instructions which implement the DES cryptographic mechanism). However, uncertified software versions of DES are common.

The security of DES has been questioned and attempts have been made to increase and improve it. Two aspects of DES relating to its security are the key length and the block size.

Doubling the block size of cipher mechanisms using multiple encryptions has been proposed in order to strengthen them. However, proposed techniques are subject to cryptanalysis.

Attempts have been made to improve DES by using multiple DES operations in sequence (using the same or different keys for each round of DES). Unfortunately, double DES (using DES to again encipher a message already encrypted by DES) has been shown to be barely stronger than DES itself. When a known plaintext-ciphertext pair is available, a search of all possible keys for the input operation, and all keys for the output operation will break the system. It has been shown that if DES has certain mathematical properties (namely if DES is a group) then cryptanalysis of DES (that is, breaking DES) would be easier than if it does not have those properties. Further, if DES has those mathematical properties, then multiple encryption would be useless and would actually weaken DES. To date it has not been shown whether DES has those mathematical properties (i.e., whether DES is a group) but most research points away from it having them. Assuming DES is not a group, the result of triple-DES (passing a message through DES three times, with the same or different keys) may be much harder to break than that of DES or double-DES. However, triple-DES is expensive and complex, and may require three hardware DES implementations and interconnecting hardware, or three software DES executions. In general multiple encryption using the same encryption technique and key does not affect the complexity of a brute-force cryptanalysis. Multiple keys are needed to enhance security.

Many other block ciphers have been developed and some of these may eventually overtake or replace DES in usage. The most likely successor to DES at present is a block cipher called IDEA. IDEA operates on 64-bit plaintext blocks using a 128-bit key (more than twice the effective key length of DES). Current software implementations of IDEA are about as fast as DES. A hardware implementation of IDEA encrypts data at a rate of 177 Mbits/second when clocked at 25MHz. The security of the IDEA cipher is not yet known, and even if it is adopted in some areas, it will be some time before it overtakes DES in use.

In general, the displacement of a commonly-used cipher by a new cipher will not take place until the community of users is satisfied that the new cipher is more secure than the old one. Since ciphers are often implemented by hardware cipher mechanisms, a change in ciphers can be expensive and may require extensive changes to a system employing them. In a large system, with some cryptographic mechanisms implemented in hardware, some in software, and others in hybrid form, changing or updating a cipher is not only costly, but potentially error prone. For example, in the communication system **120** of **FIGURES 3(a)-3(b)**, each of the cryptographic mechanisms **100** in every communication device **122** would have to be replaced. Or in a computer network employing computers **132** as in the computer system **130** of **FIGURE 3(c)**, the cryptographic mechanisms **100** on each computer **132** in the system would have to be replaced. On the other hand, when there are widespread suspicions of weakness of an existing highly used cipher, it would be useful to enhance existing systems which use that cipher.

SUMMARY OF THE INVENTION

This invention provides enhanced cryptographic mechanisms employing combinations of balanced, size-preserving block mixers, arrays of substitution mechanisms, and cipher mechanisms. When cipher mechanisms are used, the enhanced cryptographic mechanisms enhance the cipher mechanisms. The cipher mechanisms can be DES or IDEA mechanisms, or any other block cipher mechanisms, including embodiments of the present invention, and the enhanced cryptographic mechanisms increase the strength of those ciphers.

Cipher mechanisms are enhanced by having their block size increased. This increase is achieved using balanced block mixers or arrays of substitution mechanisms to spread an input to the cryptographic mechanism into multiple cipher mechanisms. Cipher mechanisms are also enhanced by transforming their inputs and outputs using balanced block mixers and arrays of substitution mechanisms.

In another aspect, this invention is a key-balanced, size-preserving block mixer for mixing two input blocks to produce two balanced output blocks, and a method of making a balanced block mixer. A balanced block mixer has two combiners, each of which combines both of the input blocks to the balanced block mixer into one output block of the mixer, such that: (i) every possible input block produces a different output block, and every possible output block is produced by a different input block; (ii) each output block is a function of both input blocks; (iii) any change to any one of the input block values changes both of the output block values; and (iv) stepping either of the input blocks through all possible values while keeping the other of the input blocks fixed steps each of the output blocks through all possible values. The method of making balanced block mixers with these properties uses techniques of constructing orthogonal Latin squares.

In another aspect, this invention is a multiple-size cryptographic mechanism which has a number of cryptographic mechanisms of different sizes and means for selecting among these cryptographic mechanisms in order to process a message. The selecting means decides which cryptographic mechanisms to use so as to minimize wasted block space in the output message.

BRIEF DESCRIPTION OF THE DRAWINGS

These and other objects and advantages of this invention will become more apparent and more readily appreciated from the following detailed description of the presently preferred exemplary embodiments, taken in conjunction with the accompanying drawings, of which:

FIGURE 1 depicts a cryptographic mechanism according to the present invention;

FIGURE 2 is a schematic diagram of an implementation of the cryptographic mechanism of **FIGURE 1**;

FIGURE 3(a) depicts a communication system employing a cryptographic mechanism according to the present invention;

FIGURE 3(b) is a schematic diagram of a communication device of the communication system of **FIGURE 3(a)**;

FIGURE 3(c) is a computer system employing a cryptographic mechanism according to the present invention;

FIGURE 4(a), **FIGURE 4(b)**, **FIGURE 5(a)**, **FIGURE 5(b)**, **FIGURE 6**, **FIGURE 7**, and **FIGURE 8** are schematic diagrams of cryptographic mechanisms which employ block mixers according to the present invention, in conjunction with block ciphers;

FIGURES 9(a) and 9(b) are schematic diagrams of balanced block mixers according to the present invention;

FIGURE 10 is a schematic diagram of a cryptographic mechanism employing substitution mechanisms according to the present invention;

FIGURE 11 is a schematic diagram of a substitution mechanism according to the present invention;

FIGURE 12, **FIGURE 13**, **FIGURE 14**, **FIGURE 15(a)**, **FIGURE 15(b)**, **FIGURE 16**, and **FIGURE 17** are schematic diagrams of cryptographic mechanisms employing substitution mechanisms according to the present invention; and

FIGURE 18 is a schematic diagram of a multiple block-size cryptographic mechanism according to the present invention.

To Part B:

Balanced Block Mixers for Block Cipher Cryptography

Efficient, Flexible, Guaranteed Mixing

A *Ciphers By Ritter* Page

Terry Ritter

The usual *balanced block mixer* is a two-input-port two-output-port mechanism which *guarantees* that any change on *one* input port will produce a change on *both* output ports. This property is sufficient to *guarantee* the overall diffusion needed in wide block ciphers.

Balanced Block Mixers are essentially two orthogonal Latin square combiners, and have both **computational** and **table look-up** realizations. (They were originally called *Block Mixing Transforms*.) There is even a **hybrid** realization which will greatly reduce table storage, at the cost of some added computation in each access.

Contents

- Overview
- Definition
- Example 1

  By Computation: X = 3A + 2B, Y = 2A + 3B
- Proof of Example 1

  Inverse, Function, Change Propagation, Balance
- Example 2

  By Table Look-Up
- Proof of Example 2

  Inverse, Function, Change Propagation, Balance
- Use

  64-Bit Block Cipher, 32-Bit Block "Cipher", Another 64-bit Block Cipher, Fenced DES
- Hybrid Realization

Overview

In the construction of block ciphers, it is necessary to use components which are far smaller than the desired block size. This makes it necessary to somehow *mix* the data from each part so that changes in any part will produce the same statistical effect in the overall block. One advantageous component for such mixing is the Balanced Block Mixer.

Definition

A Balanced Block Mixer is an *m*-input-port *m*-output-port mechanism with the following properties:

1. The mapping is one-to-one: Every possible input value (across all input ports) to the mixer produces a different output value (across all output ports), and every possible output value is produced by a different input value.
2. Each output port is a function of all input ports.
3. Any change to any one of the input ports will produce a change to every output port.
4. Stepping any one input port through all possible values (while keeping the other inputs fixed) will step every output port through all possible values.

A Balanced Block Mixer can be seen as two cryptographic combiners which are "orthogonal" (thus allowing their output to be transformed back to the original values). Each of these combiners provides "mixing" between whole blocks, with a good statistical balance between the inputs. Any particular value on an output port can be produced by any possible value on an input port, given some value on the other input port.

The properties of each part of a Balanced Block Mixer appear to be *a generalization* of additive combining like exclusive-OR. The overall structure of a Balanced Block Mixer is that of an orthogonal pair of Latin squares, which may be implemented either by computation when and as needed, or as explicit look-up tables.

Example 1

If we have two input ports labeled A and B, two output ports labeled X and Y, and some irreducible mod 2 polynomial p of degree appropriate to the port size, a Balanced Block Mixer is formed by the equations:

```
X = 3A + 2B (mod 2)(mod p),
Y = 2A + 3B (mod 2)(mod p).
```

This particular Balanced Block Mixer is a self-inverse, and so can be used without change whether enciphering or deciphering. One possible value for p is 100011011 in binary (this is an irreducible mod 2 polynomial).

An unusual advantage of the computational realization is *scalability*. We can make a proper Balanced Block Mixer of any reasonable size which requires only linear computational effort with port size. Not only can we use very large Balanced Block Mixers, we can also use arrays of smaller Balanced Block Mixers for about the same overall cost. This turns out to be very useful, as this sort of mixing can be repeated with blocks of decreasing size to provide guaranteed mixing to increasingly fine block elements.

Proof of Example 1

ASSERTION: (We have a finite field.) Mod-2 polynomials modulo some irreducible polynomial p generate a finite field.

(Comment: Proofs can use algebra.)

PROPOSITION: (Example Balanced Block Mixer.) The equations

```
X = 3A + 2B = A + 2(A + B)
Y = 2A + 3B = B + 2(A + B)
```

and the inverse

```
A = X + 2(X + Y)
B = Y + 2(X + Y)
```

mod 2 and mod p, where p is some irreducible mod 2 polynomial, represent a Balanced Block Mixer.

1. **Inverse Proof:** Substitute the formulas for X and Y into the formulas for A and B:

   ```
   A = A + 2(A + B) + 2(A + 2(A + B) + B + 2(A + B))
   A = A + 2(A + B) + 2(A + B) = A
   ```

   and

   ```
   B = B + 2(A + B) + 2(A + 2(A + B) + B + 2(A + B))
   B = B + 2(A + B) + 2(A + B) = B
   ```

   so the inverse does exist for any polynomials A and B. And a function with an inverse on the same range must be one-to-one.
2. **Function Proof:** the equations for output code X includes both input code values A and B, so X is a function of both input codes. Y reasons similarly.
3. **Change Propagation Proof:** First consider one term of one output block equation:

   Suppose some change C is added to A:

   ```
   X  = 3A + 2B  (mod 2, mod p)
   X' = 3(A+C) + 2B
   X' = 3A + 3C + 2B
   dX = X' - X = 3C
   ```

   So, for any non-zero change C, we have a different X. Similar reasoning covers the other term, and the other equation.)
4. **Balance Proof:** Suppose that stepping an input through all possible values does *not* step an output through all possible values. Since the input and output blocks are the same size, some output value must occur for a plurality of input values. Assuming A is fixed, there must be at least two different values, B and B', which produce the same X:

   ```
   X = 3A + 2B = 3A + 2B'
   ```

   so

   ```
   X + 3A = 2B = 2B'
   ```

   which implies that

   ```
   B = B'
   ```

   which is a contradiction. Fixing B or working on the other block reason similarly.

A consequence of this particularly efficient construction is that this Balanced Block Mixer has essentially no "strength" of its own. As one example of mixing without strength, consider DES with a known key: this is also a "weak" operation, but would nevertheless provide good, invertible mixing for two 32-bit input blocks.

Example 2

If we have two input ports labeled A and B, two output ports labeled X and Y, and orthogonal Latin square arrays Q and R of order appropriate to the port size, a Balanced Block Mixer is formed by the equations:

```
X = Q[A][B],
Y = R[A][B].
```

where Q[A][B] signifies indexing the stored array Q by row and column.

When we know Q and R, we can construct inverse arrays Q' and R' which take X and Y back to A and B.

One advantage of the table look-up realization is that the squares can be very random-like, nonlinear, strong and keyed. And even though the table realization is not scalable, these relatively small mixings can be extended to cover large blocks with Fast Fourier Transform (FFT)-style mixing patterns. The simple storage-oriented structure of this realization seems ideal for modern chip implementations.

Proof of Example 2

DEFINITION: (Square Array.) A square array of order *n* is a two dimensional array of *n*n* storage elements indexed by row and column. Each possible combination of row and column values will select a storage position which can hold a single symbol or value.

DEFINITION: (Latin Square Array.) A Latin square array of order *n* contains each of *n* unique symbols or values *n* times, such that each possible value occurs exactly once in each row and column.

DEFINITION: (Array Pair.) Two square arrays of order *n* can form an ordered pair of values when each square is indexed with the same row and column values.

DEFINITION: (Orthogonal Array Pair.) An array pair is said to be *orthogonal* when all *n*n* possible row and column index values produce every possible ordered pair exactly once.

COROLLARY: (Invertibility.) An orthogonal array pair is always invertible, since no ordered pair value occurs more than once.

PROPOSITION: (Example Balanced Block Mixer.) Two orthogonal Latin squares Q and R of appropriate order with array indexing:

```
X = Q[A][B],
Y = R[A][B].
```

form a Balanced Block Mixer. Q[A][B] signifies indexing the stored array Q by row and column,

1. **Inverse Proof:** Because Latin squares Q and R are orthogonal, each the ordered pair (X,Y) is unique, so each inverse transformation (taking X,Y to A,B) is also unique.
2. **Function Proof:** Because Latin squares Q and R are indexed by both A and B, both X and Y are functions of both indexes.
3. **Change Propagation Proof:** First consider one combiner:

   Suppose some change C is added to A:

   ```
   X  = Q[A][B]
   X' = Q[A+C][B]
   ```

   For any non-zero change C, the new index A+C selects a different entry within column B. Since each entry in one column of a Latin square is unique, X' is necessarily different than X.

   Similar reasoning covers the column index B, and the other combiner R.
4. **Balance Proof:** Suppose we step input A through all possible values, leaving B fixed. This will traverse all entries in one column in each square (Q and R). Since a Latin square has exactly one occurrence of each value in every column, we necessarily traverse each possible output value in both X and Y.

   Stepping the other index (B) reasons similarly.

One consequence of the table realization is the ability to explicitly specify strong, nonlinear Latin squares. The need to store the table is a cost, but table access can be substantially faster than computation.

It is probably easiest simply to consider a Balanced Block Mixer to *be* a pair of orthogonal Latin squares, realized either by computation or explicit values in a look-up table.

Use

Any really new component seems strange when first proposed, because there is no background showing such a component is useful. For example, transistors were difficult to understand and use when they were new, and at that time the need for an amplification component was already well established. While Balanced Block Mixers are not the next transistor, they do have certain advantages which are worth taking seriously.

Computational-based realizations of Balanced Block Mixers like that of Example 1 are typically weak. A question that immediately arises is whether there can be any advantage to using a *weak* component in a cipher. But, by itself, exclusive-OR is also *weak* (having no strength at all), and is used in ciphers all the time without similar questions.

Table-based realizations of Balanced Block Mixers like those in Example 2 are necessarily limited to small block widths. The obvious question here is whether there can be any advantage to using a *small* mixing component. But even a small mixing component can be used in FFT-like patterns, and this *does* support the mixing of large blocks.

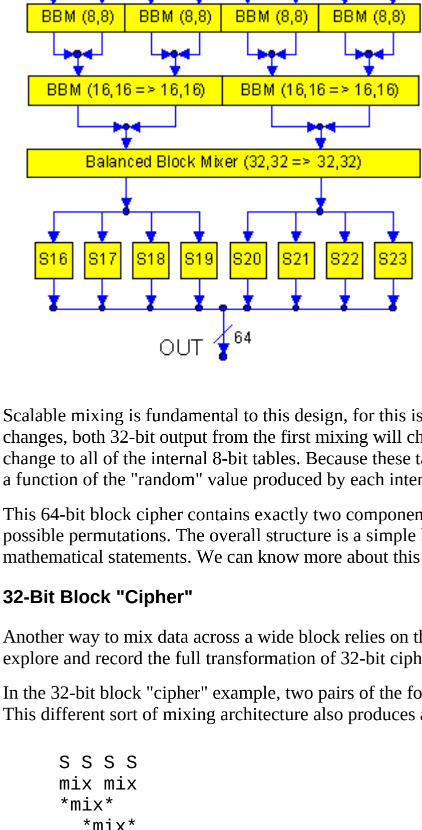
The advantage of the Balanced Block Mixer lies in the properties which the component is guaranteed to possess. In particular, any change at all to one input port value is *guaranteed* to change both output port values. The use of this simple property is sufficient to guarantee overall diffusion or "avalanche" in a block cipher.

64-Bit Block Cipher

One example of the practical use of Balanced Block Mixers is a 64-bit block cipher based on 8-bit substitution tables:

```
S S S S S S S S
------mix------
--mix-- --mix--
mix mix mix mix
S S S S S S S S
mix mix mix mix
--mix-- --mix--
------mix------
S S S S S S S S
```

The 64-bit input is split and enters eight 8-bit substitution tables. Then we have a single mixing on the two 32-bit-wide values accumulated from four tables each. Then we have two mixings on the four 16-bit-wide values across the block, and four mixings on the eight 8-bit-wide values across the block. This result is again substituted through eight substitution tables, and then mixed again in three layers in opposite order, before being substituted a last time in eight 8-bit substitution tables. In practice, each table would be initialized by individual shuffling according to a cryptographic key.



Scalable mixing is fundamental to this design, for this is how we can achieve diffusion across the whole 64-bit block while using tiny 8-bit-wide confusion components. If even one bit of the input block changes, both 32-bit output from the first mixing will change, as will all four 16-bit results and then all eight 8-bit results from mixing. Accordingly, Balanced Block Mixing has provably conducted the change to all of the internal 8-bit tables. Because these tables are all shuffled or "keyed," each responds with a "random" value. These values are then mixed again, and we can guarantee that the result is a function of the "random" value produced by each internal table.

This 64-bit block cipher contains exactly two component types, each of which is susceptible to deep mathematical analysis. The substitution tables are each simply arbitrary selections among all possible permutations. The overall structure is a simple logical architecture, and far simpler than any 16-round cipher. This leads to a cipher about which we can actually begin to make serious mathematical statements. We can know more about this type of cipher than we can about most others.

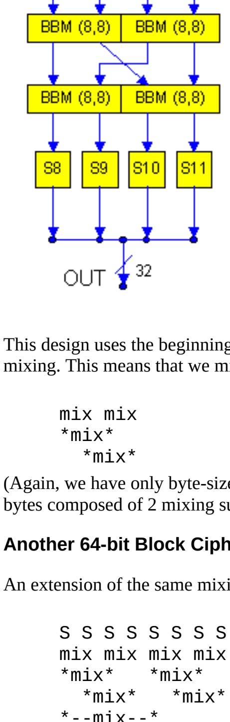
32-Bit Block "Cipher"

Another way to mix data across a wide block relies on the fast, perfectly-balanced mixing in Balanced Block Mixers. Another example is a 32-bit block "cipher." (Note that it is certainly possible to explore and record the full transformation of 32-bit cipher, no matter what the keyspace of the cipher is. Thus, a 32-bit block cipher must be considered "weak.")

In the 32-bit block "cipher" example, two pairs of the four substitution table outputs are mixed into two element pairs. One element of each of these pairs is then mixed with an element of the other pair. This different sort of mixing architecture also produces a result in which each element is a function of every input element.

```
S S S S
mix mix
*mix*
  *mix*
S S S S
  *mix*
*mix*
mix mix
S S S S
```

Here we have only byte-sized mixing, and "*mix*" means to mix the two bytes in the columns covered by "mix" or which have "*" symbols.



This design uses the beginning of FFT-style mixing patterns. While it is set up to use keyed substitutions to strengthen and hide weak linear mixing, it could instead use strong, keyed, table look-up mixing. This means that we might be able to eliminate the need for keyed substitutions entirely:

```
mix mix
*mix*
  *mix*
```

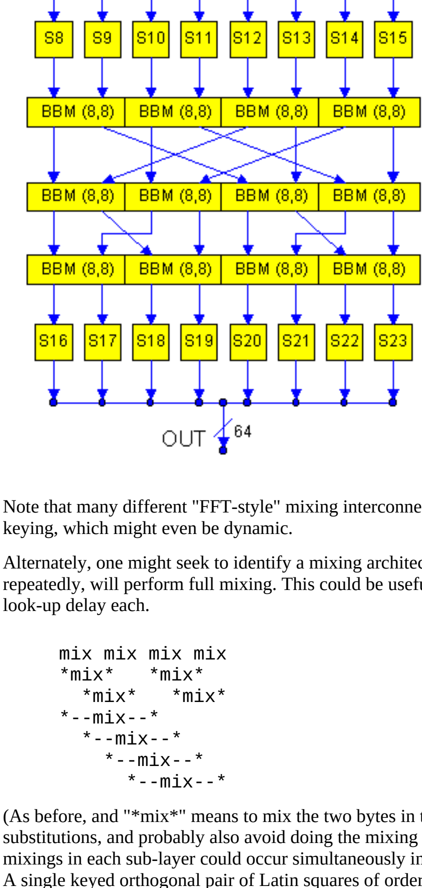
(Again, we have only byte-sized mixing, and "*mix*" means to mix the two bytes in the columns covered by "mix" or which have "*" symbols.) This leaves us with one overall mixing from 4 bytes to 4 bytes composed of 2 mixing sub-layers of 2 mixings each.

Another 64-bit Block Cipher

An extension of the same mixing concept produces a 64-bit block cipher (and could be extended to an arbitrary power-of-2 number of substitution- or cipher-width elements).

```
S S S S S S S S
mix mix mix mix
*mix*   *mix*
  *mix*   *mix*
*--mix--*
  *--mix--*
    *--mix--*
      *--mix--*
S S S S S S S S
*--mix--*
  *--mix--*
    *--mix--*
      *--mix--*
*mix*   *mix*
  *mix*   *mix*
mix mix mix mix
S S S S S S S S
```

(Again, we have only byte-sized mixing, and "*mix*" means to mix the two bytes in the columns covered by "mix" or which have "*" symbols.) Like the previous example, two mixing layers produce sets of four elements, each of which is a function of four input elements. Then, just one more mixing step mixes the two sets of elements (in any of 24 possible permutations per mixing layer) to two sets of results, each element of which is a function of all eight original input elements. (The mixing pattern is somewhat different so it can be shown easily; many mixing patterns are available, and equally powerful.)



Note that many different "FFT-style" mixing interconnections can mix all of the input values into each of the output values. Selecting a particular mixing architecture could provide some amount of keying, which might even be dynamic.

Alternately, one might seek to identify a mixing architecture which has an especially fast implementation. In particular, it is possible to have a fixed interconnection permutation which, when used repeatedly, will perform full mixing. This could be useful in a hardware table realization which uses one row of Balanced Block Mixers repeatedly, and would have only *log n* rounds of one table look-up delay each.

```
mix mix mix mix
*mix*   *mix*
  *mix*   *mix*
*--mix--*
  *--mix--*
    *--mix--*
      *--mix--*
```

(As before, and "*mix*" means to mix the two bytes in the columns covered by "mix" or which have "*" symbols.) But if we do use keyed table realizations, we probably can avoid using the substitutions, and probably also avoid doing the mixing twice. Here we have 3 mixing sub-layers of 4 mixings each which performs a thorough, keyed, nonlinear and independent mixing. Each of the 4 mixings in each sub-layer could occur simultaneously in separate hardware. Strength issues probably depend upon the number of keyed mixers used and possible dynamic changes in the mixing pattern. A single keyed orthogonal pair of Latin squares of order 256 at least potentially has a keyspace of 3368 bits.

Fenced DES

The original application for Balanced Block Mixing technology was to expand the block width of existing block ciphers, such as DES. This application used the original multiple-size mixing approach. By replacing substitution components with entire ciphers, and resizing the mixers to the new element size, we get a new, larger cipher which uses a smaller cipher as a component. This approach is described in the Fenced DES design documents.

Hybrid Realization

It is possible to combine some of the advantages of both computational and table realizations.

- A **computational** implementation uses linear operations, which means that no internal table storage is needed, but also means that it must have some form of external "protection." This is often a "fencing" array of keyed substitutions, which *does* require table storage.
- A **table look-up** implementation does not need external protection, but does need substantial internal tables. For example, a byte-wide Balanced Block Mixer would use two orthogonal Latin square tables of 64KB each. Of course, this is much less of a problem than it would have been even 5 years ago.
- A **hybrid** implementation could use two small byte-substitutions to permute the row and column values used by a byte-size computational BBM. The result would be a nonlinear byte-width block mixer using only 1/4 KB of storage, although it *would* require computational delay beyond table look-up. It also could be applied in FFT-like patterns to cover a wide block.

Consider the design of a 256-byte mixing large-block cipher (the 2048-bit block may allow us to avoid "operating modes" like CBC, and would have room for options like authentication and dynamic keying). Now, we could use just a single BBM and step across byte-pairs until we have done them all; this would involve 128 look-up and computation delays for each of 8 mixing sub-layers.

Alternately, we could also use 128 separate byte BBM's in hardware, and do 128 mixings in parallel, in one look-up and computation delay. With 128 hybrid byte BBM's we need only 64KB of store. This is 1/2 the store used in *one* table look-up realization.

We will need 8 mixing sub-layers, which we can either implement with more hardware, or by latching a permuted result from each previous mixing. (We should probably stay away from using the term "rounds" for such operation, because of its association with Feistel structures.)

In this way we perform a complete nonlinear mixing of 256 bytes into 256 bytes in 8 look-up and computation delays. This may be faster than we could get the data to and from the ciphering engine.

Terry Ritter, his current address, and his top page.

Last updated: 1997-01-09

Active Balanced Block Mixing in JavaScript

High-quality mixing makes it difficult to distinguish and attack individual components.

A *Ciphers By Ritter* Page

Terry Ritter

Introduction

A Balanced Block Mixer can be seen as a pair of orthogonal Latin squares. Each square is used as a combiner to mix two input values. In a Latin square combiner, any output value can be produced by any possible value on one input, by placing some appropriate value on the other input. This is balance under all conditions, and is a generalization of the protection in common additive combiners like exclusive-OR. The advantage of orthogonal combiners is that the result is invertible: the mixing can be reversed using only the mixed result. There can be no orthogonal exclusive-OR's.

Nonlinear BBM's

It is certainly possible to create and key large explicit Latin square tables. But even an "8-bit" combiner has 65,536 byte entries, and there would be two of those, for a total of 128K bytes. This is both larger and more of a keying delay than we might like. Consequently, the linear form (which needs no keying) seems worth exploring.

Linear BBM's

The BBM form I often use can be expressed in two equations:

```
X = 3A + 2B (mod 2)(mod p),
Y = 2A + 3B (mod 2)(mod p).
```

Again, this is a linear mixing, with **no strength at all**. Its advantage is the ability to combine two block values (here A and B) into two other block values (here X and Y) which *both* depend upon *each* of the input values, and do so in a perfectly balanced way. Yes, these results are linearly related to the inputs, **but** if we add four keyed substitution tables, one on each input and each output, we have a brand new situation. (Keying the tables is straightforward; see, for example: **A Keyed Shuffling System**, on these pages.) It is not at all clear how one can approach an analysis of such a system. But this structure is scalable, and we can scale it down, down, down, until we ought to be able either to attack it by hand, or know the reason why not. This is, of course, the truth we seek.

BBM's in Mod 2 Polynomials

One of the problems understanding these simple linear BBM's may be that they are based on mod 2 polynomials. While these are actually simpler than integers and conventional arithmetic, they will be new to many people. Basically we are just generating a mathematical field with 2^n values which supports a form of addition and multiplication which produce values in that same field.

For example, suppose we have two 2-bit values to be combined. With 2-bit values, we only have a value range from 0 through 3, so lets choose 1 (01) and 2 (10), with the irreducible polynomial 7 (111).

First we add A and B, which is just a bit-by-bit exclusive-OR:

```
  01   A
+ 10   B
  --
  11   A+B (mod 2)
```

Next we multiply by 2, which is just a left shift:

```
 110   2(A+B) (mod 2)
```

But now the leftmost bit is set, which means the result is out of range. To bring it back in, we subtract p. Again, this is just a bit-by-bit exclusive-OR:

```
 110   2A+2B (mod 2)
- 111   p
 ---
  01   2A+2B (mod 2)(mod p)
```

Then we finish off the two equations:

```
  01   2A+2B (mod 2)(mod p)
+ 01   A
  --
  00   3A+2B (mod 2)(mod p) = X = 0


  01   2A+2B (mod 2)(mod p)
+ 10   B
  --
  11   2A+3B (mod 2)(mod p) = Y = 3
```

And, if we enter 1 and 2 below (with "2 Bit" selected), we get 0 and 3 as a result:

Active Balanced Block Mixer

Select Mixing Polynomial (by Element Bit Width)

2 Bit: ○ 111
3 Bit: ○ 1011 ○ 1101
4 Bit: ○ 10011 ○ 11001 ○ 11111

Enter Hex Values

Balanced Block Mixing		
Input		
Output		

Operating the Active BBM

In the active Balanced Block Mixer above, we can place values (within the appropriate range) in the top entry fields, and get the mixed results in the bottom fields. As each value is entered, the results are updated (hit "Enter," tab to the next field, or click elsewhere to enter the value). We can also put any results we have in the bottom entry fields, which will update the inputs, and so show what inputs mix to that result.

Another way to think of the mixing is as the overall discrete transformation in the form of two explicit orthogonal Latin squares. For a particular element width and polynomial, we can make a table containing both squares by pressing the "Make Table" button. In the table, the left mixing input selects a row, the right input a column, thus selecting a particular entry. Each entry has two digits: one from the left square, and one from the right. In the "2 Bit" table, selecting row 1 and column 2 gives 03 which is the result we saw before.

Mixing 4 Elements

If we could mix together only two elements, the BBM concept would not be very helpful. But we *can* mix more elements, any power-of-2 elements, to be exact. We do this by mixing two elements at a time, and then mixing those results with other mixed results. This is the old FFT strategy, which results in mixing n elements in n log n time.

The next active mixing combines 4 elements at a time. Note that changing even a single input value always changes all 4 of the output values. This is not the binomial distribution we expect from avalanche. But when each of the output values is translated through a keyed substitution table, the bit-change statistics improve nicely. And the real issue is whether the high quality of the mixing used to combine multiple substitution tables prevents those tables from being attacked individually.

Active 4-Element Balanced Block Mixing

Enter Hex Values or Click Buttons

Operating the Active 4-Element Balanced Block Mixing

Enter the desired value in an input, or an output (remember to use "Enter," or tab to the next field, or click outside the box to enter each value), or just click the "-" or "+" buttons to change the values. This mixing uses the bit-width and polynomial selected above, and so can handle 2-bit, 3-bit or 4-bit elements.

Note how changing any single input value -- even by just a single bit -- changes *all* of the output values. This is the basis for a guarantee that even a single bit-change absolutely will propagate to *each* and *every* subsequent element. So if we have a keyed substitution table on each output, we are assured that even a single input bit-change will engage the strength of each table, which will also produce a good bit-change distribution result.

This is an invertible system, so it is obviously possible to supply some input change which will limit the output change to just a single element. (Just change one of the output values to see what the input would have to be.) But if we have keyed tables on the input, it would seem to be difficult to exploit the mixing linearity we know is there. This is the basis of Mixing cipher design.

***Terry Ritter**, his **current address**, and his **top page**.*

***Last updated:*1998-03-29**

Newsgroups: sci.crypt
Path: cactus.org!ritter
From: ritter@cactus.org (Terry Ritter)

Subject: Block Mixing Transformations
Message-ID: <1994Mar13.051515.27175@cactus.org>
Keywords: DES replacement, Large blocks
Organization: Capital Area Central Texas UNIX Society, Austin, Tx
Date: Sun, 13 Mar 1994 05:15:15 GMT

Ritter Software Engineering
2609 Choctaw Trail
Austin, Texas 78745
(512) 892-0494, ritter@cactus.org

Keyed Balanced Size-Preserving Block Mixing Transforms

Terry Ritter
March 12, 1994

Introduction

Modern block ciphers seek to emulate extremely large substitution tables algorithmically, using complex combinations of various simple internal mechanisms. These internal mechanisms include small substitutions and trivial combinings, but the art and mystery of block cipher design is how to couple these simple and weak operations in ways which produce a strong overall cipher.

One apparently new type of mechanism which might be useful in block cipher design would take two blocks in, share data between them, and then produce two generally-different blocks as a result. In particular, this mechanism might be used to mix data to (and from) a pair of substitutions, thus hopefully producing a stronger result than the two substitutions operating separately and independently. In most cases, it would be necessary for the mechanism to have an inverse, and to produce output blocks of the same size as the input. The result would be a mechanism which could be inserted anywhere in the internal data paths common in block-cipher designs.

Block Mixing Transforms

Consider constructs like this:

```
          A              B
          |              |
          v              v
          Mixing Transform
          |              |
          v              v
          X              Y

          X              Y
          |              |
          v              v
          Inverse Transform
          |              |
          v              v
          A              B
```

Capital letters represent data blocks. Alternately, we can describe the transform, in general, as:

```
     X := f1( A, B );   Y := f2( A, B );

     A := f3( X, Y );   B := f4( X, Y );
```

The intent of such a system is to mix two input blocks in a complex yet reversible way. This could provide two advantages:

1) It should make each output bit a function of all the input bits (on average), thus providing a way to expand block size while using smaller block-cipher functions. Hopefully the construct would also defeat attempts to "divide-and-conquer" the smaller functions separately.

2) It could provide a way to connect block-cipher functions in sequence, while eliminating any fixed direct connection between the blocks, such connections being vulnerable to "fix-in-the-middle" attack.

A mixing transform is not unlike a "butterfly" section in a fast Fourier transform (FFT) [3]. But the usual FFT operates on complex values, and these are normally represented in floating-point. When implemented in fixed-point (as needed for mixing data blocks), the normal FFT butterfly expands the range of the input values, thus requiring a larger amount of storage (a larger block size) for the result. Fast Hadamard / Walsh transforms [2] behave similarly.

For cryptography, we need transforms which are "size preserving" so that we can perform fixed-size block operations (such as DES) either on the input data or on the transformed results. It was not clear to me that this was going to be possible (at least with equations of practical complexity) until Eli Biham provided some examples of size-preserving mixing transforms:

```
     X := A - B;   Y := 2A - B;

     A := Y - X;   B := Y - 2X;
```

for n-bit blocks, A, B, X, and Y, and arithmetic mod 2^n.

There are actually many such transforms, and Biham has found a generalized form:

```
     (-1  1 )
     (-w  w-1)
```

and

```
     (w-1  -1)
     (w    -1)
```

where w is some constant. For example, when w = 2:

```
     X := -1*A +     1*B  =  B -  A
     Y := -2*A + (2-1)*B  =  B - 2A

     A := (2-1)*X + -1*Y  =   X - Y
     B :=     2*X + -1*Y  =  2X - Y
```

with the arithmetic mod 2^n.

To see inverse, note that

```
     A =   X - Y  =   (B - A) - (B - 2A)  =  A
     B =  2X - Y  =  2(B - A) - (B - 2A)  =  B
```

These are fixed, linear transformations. If we know the input values, and the transformation, we will also know the output values. Even when the full equation is unknown, the simplicity and linearity of these transforms means that they require special protection in cryptographic applications. Mixing transforms can only be used when both the input and the output values cannot be exposed simultaneously.

Alas, the transform mentioned above has a problem: Specifically, the least-significant-bit (lsb); that is, lsb(Y) = lsb(B). This is because the expression B - 2A has shifted A left one bit, leaving the bottom bit of B exposed. This provides a bit of direct correlation between an input value and an output value. This is probably sufficient to support a practical "fix-in-the-middle" attack if the transform is used to isolate two DES operations.

Consider these correlation experiments on the above transform with 4-bit blocks:

```
         x3  x2  x1  x0  y3  y2  y1  y0

     b0  64  64  64  64  64  64  64 128
     b1  64  64  64  64  64  64  64  64
     b2  64  64  64  64  64  64  64  64
     b3  64  64  64  64  64  64  64  64
     a0  64  64  64  64  64  64  64  64
     a1  64  64  64  64  64  64  64  64
     a2  64  64  64  64  64  64  64  64
     a3  64  64  64  64  64  64  64  64
```

This is a 0 -> 0 correlation count. For each possible input value (over both A and B), for each input bit which is zero (somewhere in A and B) and each output bit which is zero (somewhere in X and Y), a count is recorded. The count of 128 means that y0, the lsb of Y, occurs twice as often as expected when the lsb of B is zero.

Similarly,

```
         64  64  64  64  64  64  64   0
         64  64  64  64  64  64  64  64
         64  64  64  64  64  64  64  64
         64  64  64  64  64  64  64  64
         64  64  64  64  64  64  64  64
         64  64  64  64  64  64  64  64
         64  64  64  64  64  64  64  64
         64  64  64  64  64  64  64  64
```

a 0 -> 1 correlation count, shows that no cases exist where the lsb of B is a one and the lsb of Y is a zero.

Cryptographic Mixing

In [8] I introduced a new type of reversible stream-cipher combiner (the first stream-cipher combiner, which we now call "exclusive-OR" or "mod-2 addition" was described by Vernam [12]). "Combiner" is the traditional cryptographic name for a mixing function. [11,5,1] (Non-reversible combiners are also used, typically to make confusion sequences difficult to penetrate. [e.g., 6]) Combiners and mixing transforms have much in common.

Basically, a combiner will look like any other two-input one-output function:

```
          A            B
          |            |
          v            v
          Mixing Function
          |
          v
          C

          C             B
          |             |
          v             v
          Inverse Function
          |
          v
          A
```

The capital letters represent the block size; in a typical stream cipher these are byte values. A is the plaintext, B the confusion stream, C the ciphertext. Note that exactly the same confusion stream is needed to recover the original data; this is the heart of stream-cipher security.

There are many two-input functions, but most are not useful as cryptographic data combiners, which must be reversible and must have no correlation between either input and the output. Combiners which do have correlation [e.g., 4] fall to statistical attacks [e.g., 10]. If we see mixing transforms as a matched-set of cryptographic combiners, we can see that correlation is a problem with the example transform. (Biham did have an example of one balanced but non-keyed transform based on rotation and subtraction mod 2^n.)

Mixing in Mod-2 Polynomials

Since the "weak" exclusive-OR form of combiner has long been available, modern combiner designs are normally intended to be "stronger" and, thus, are more complex. But it is not at all clear that "stronger" is what we need in a mixing transform. Presumably, "strength" can be provided more efficiently by some other function, like DES, or a substitution table. Thus, we may really want a modest-strength extremely-fast mixing solution, and one approach is to consider the well-known field of mod-2 polynomials.

In mod-2 arithmetic, addition is the same as subtraction

```
     X + Y  =  X - Y
```

and any value added to itself is zero

```
     X + X = 0
```

so, in general, multiplication cannot be achieved by addition

```
     X + X <> 2X
```

(assuming X is non-zero) but is instead achieved by shifting. Then

```
     2X + X = 3X
```

so multiplication is not restricted to binary powers. Of course

```
     3X + X = 2X
```

which just shows that mod-2 arithmetic can be surprising.

It is interesting to see just how unusual good mixing transforms are. Consider a first approach

```
     X := A + B;  Y := A - B;
```

(mod-2, mod-p, where p is some primitive mod-2 polynomial of appropriate degree for the size of the data blocks). While this is a reasonable approach in the integers, in mod-2 polys, A + B = A - B. This means that X = Y, and the two resulting identical blocks cannot possibly carry enough information to provide an inverse transform for two arbitrary input blocks. It does not work.

Next consider

```
     X := A + B;  Y := A + 2B;
```

with inverse operations

```
     A := (2X + Y) / 3;   B := (X + Y) / 3;
```

(mod-2, mod-p), and the division done by multiplying by the inverse of 3, mod p. (Appropriate inverse equations may not always exist; finding the inverse equations is interesting in itself.) This works. But here X is never affected by p at all, thus producing an extremely regular (and un-keyed) transformation. And the inverse multiplication is, in general, far more expensive than multiplication by a small integer.

Finally, consider

```
     X := 2A + 3B;   Y := 3A + 2B;

     A := 2X + 3Y;   B := 3X + 2Y;
```

Again, operations are mod-2 and mod-p, where p is some primitive mod-2 polynomial of appropriate degree for the data blocks X, Y, A and B. This works, and the transform is a self-inverse. The primitive affects the result in both data blocks. And the multiplications are simple.

Correlation experiments conducted as before show a nice, balanced, uncorrelated system:

```
         64  64  64  64  64  64  64  64
         64  64  64  64  64  64  64  64
         64  64  64  64  64  64  64  64
         64  64  64  64  64  64  64  64
         64  64  64  64  64  64  64  64
         64  64  64  64  64  64  64  64
         64  64  64  64  64  64  64  64
         64  64  64  64  64  64  64  64

         64  64  64  64  64  64  64  64
         64  64  64  64  64  64  64  64
         64  64  64  64  64  64  64  64
         64  64  64  64  64  64  64  64
         64  64  64  64  64  64  64  64
         64  64  64  64  64  64  64  64
         64  64  64  64  64  64  64  64
         64  64  64  64  64  64  64  64
```

These functions are extremely fast. Addition is a simple exclusive-OR. Multiplication by two is simply a left-shift and a conditional add of the primitive. Multiplication by three is a multiplication by two plus an addition.

Keyed Mixing Transforms

The mod-2 polynomial transforms depend on having some primitive of the appropriate degree. Different primitives produce different mixing functions, with similar overall performance. This leads to the possibility of keying the transforms by selecting arbitrary primitives. (Some references to primitive-finding algorithms are given in [9].)

Rabin gives the number of degree-n primitives as about p^n / n [7]. Thus, for degree 64, we have about 2^64 / 2^6 or about 2^58 primitives. This means that each randomly-selected degree-64 primitive carries about 58 bits of key. Of course, this key can only be effective to the extent that the linear transformation cannot be attacked and the primitive thus deduced.

Some Consequences

If a single input bit changes on one of the mixing transform input blocks, we can be sure that at least one bit will change in both output blocks.

If two input bits change, we can be sure that these bits will not "cancel" each other; changes will still occur in the output blocks.

If many input bits are changed, and the transform primitive is known, it is possible to engineer a no-change in one output block (although this is unlikely to happen by chance). Should this be undesirable, it might be made impossible by design (such as ciphering the input blocks before mixing), or by keying the transform (so the necessary bit patterns are unknown).

If it becomes possible to define the input to, and what the output must be from a ciphering element, it will be possible to key-search that element independent of other elements, and this is what we hope to avoid. To prevent this it may be necessary to use keyed input and output transforms, or even multiple ciphering levels between transforms.

Applications

It is crucial to remember that these simple, high-speed, but linear mixing transforms can be said to have "strength" only if the input and output values are never both available. That is, these structures do not by themselves handle "known-plaintext" attack. (Of course, the same could be said for many other simple internal mechanisms used in block cipher construction.)

Simple constructs like

```
        A      B
        |      |
        v      v
        MixTrans
        |      |
        v      v
        C      D
```

are not likely to be very useful as ciphers by themselves, even if the mixing transformation is keyed and the blocks are large.

On the other hand, constructs like

```
        A      B
        |  p1  |
        v  v   v
        MixTrans
        |      |
        v      v
      DES1   DES2
        |      |
        |  p2  |
        v  v   v
        MixTrans
        |      |
        v      v
        C      D
```

are considerably more interesting. Note that this construct ciphers a double-size DES block at single-DES rates. It seems to require keyed mixing transforms. Similarly,

```
        A      B
        |      |
        v      v
      DES1   DES2
        |      |
        |  p   |
        v  v   v
        MixTrans
        |      |
        v      v
      DES3   DES4
        |      |
        v      v
        C      D
```

will cipher a double-size DES block at double-DES rates, and at least superficially avoids all weakness in the mixing transform by placing strength in each input and output port. This may avoid the need to key the mixing transform.

Alternately,

```
          A              B
          |      k1      |
          v      v       |
        XOR <- DES1-----|
          |              |
          |      k2      |
          |      v       v
          |---- DES2 -> XOR
          |              |
          |      p       |
          v      v       v
          Mixing Transform
          |              |
          |      k3      |
          v      v       |
        XOR <- DES3 ----|
          |              |
          |      k4      |
          |      v       v
          |---- DES4 -> XOR
          |              |
          v              v
          C              D
```

also ciphers at double-DES rates.

Of course, larger external blocks mean an increase in the number of internal data paths, making various sorts of interconnection configurations possible. Thus

```
          A      B      C      D
          |  p1  |      |  p2  |
          v  v   v      v  v   v
          MixTrans1     MixTrans2
       p3 |      |  p4  |      |
        v  v     v  v   v      v
       -Trans3    MixTrans4    Mix-
          |      |      |      |
          v      v      v      v
         DES1   DES2   DES3  DES4
          |      |      |      |
          |  p5  |      |  p6  |
          v  v   v      v  v   v
          MixTrans5     MixTrans6
       p7 |      |  p8  |      |
        v  v     v  v   v      v
       -Trans7    MixTrans8    Mix-
          |      |      |      |
          v      v      v      v
          E      F      G      H
```

will cipher quadruple-size DES blocks at single-DES rates,

```
          A      B      C      D
          |      |      |      |
          v      v      v      v
         DES1   DES2   DES3   DES4
          |      |      |      |
          |  p1  |      |  p2  |
          v  v   v      v  v   v
          MixTrans1     MixTrans2
      p3  |      |  p4  |      |
       v  v      v  v   v      v
      -Trans3    MixTrans4     Mix-
          |      |      |      |
          v      v      v      v
         DES5   DES6   DES7   DES8
          |      |      |      |
          v      v      v      v
          E      F      G      H
```

will cipher quadruple-size DES blocks at double-DES rates, and

```
          A             B              C             D
          |      k1     |              |     k2      |
          v      v      |              v     v       |
         XOR <- DES1 ----|            XOR <- DES2 ----|
          |             |              |             |
          |      k3     |              |     k4      |
          |      v      v              |     v       v
          |---- DES3 -> XOR            |---- DES4 -> XOR
          |             |              |             |
          |      p1     |              |     p2      |
          v      v      v              v     v       v
          MixingTransform1             MixingTransform2
  p3      |             |      p4      |             |
  v       v             v      v       v             v
-Transform3             MixingTransform4             Mixing-
          |             |              |             |
          |      k5     |              |     k6      |
          v      v      |              v     v       |
         XOR <- DES5 ----|            XOR <- DES6 ----|
          |             |              |             |
          |      k7     |              |     k8      |
          |      v      v              |     v       v
          |---- DES7 -> XOR            |---- DES8 -> XOR
          |             |              |             |
          v             v              v             v
          E             F              G             H
```

will also cipher quad-size blocks at double-DES rates. But in each case, four double-level mixing transforms could be replaced by a single double-size mixing transform:

```
          A      B  p1  C      D
          |      |  |   |      |
          v      v  v   v      v
          ---------mix1---------
          |      |      |      |
          v      v      v      v
         DES1   DES2   DES3   DES4
     p2   |      |      |      |
     v    v      v      v      v
     ix2---------       --------m
          |      |      |      |
          v      v      v      v
          E      F      G      H


          A      B      C      D
          |      |      |      |
          v      v      v      v
         DES1   DES2   DES3   DES4
          |      |      |      |
          |      |  p   |      |
          v      v  v   v      v
          ---------mix---------
          |      |      |      |
          v      v      v      v
         DES5   DES6   DES7   DES8
          |      |      |      |
          v      v      v      v
          E      F      G      H


          A             B              C             D
          |      k1     |              |     k2      |
          v      v      |              v     v       |
         XOR <- DES1 ----|            XOR <- DES2 ----|
          |             |              |             |
          |      k3     |              |     k4      |
          |      v      v              |     v       v
          |---- DES3 -> XOR            |---- DES4 -> XOR
          |             |              |             |
          |             |      p       |             |
          v             v      v       v             v
          ---------------------mix----------------------
          |             |              |             |
          |      k5     |              |     k6      |
          v      v      |              v     v       |
         XOR <- DES5 ----|            XOR <- DES6 ----|
          |             |              |             |
          |      k7     |              |     k8      |
          |      v      v              |     v       v
          |---- DES7 -> XOR            |---- DES8 -> XOR
          |             |              |             |
          v             v              v             v
          E             F              G             H
```

These are new ciphering architectures. Clearly, it is not known how strong these constructs would be. However, this situation can hardly be considered unusual.

Other opportunities exist when constructing completely new block ciphers. These might, for example, be based on byte-wide key-permuted substitutions, thus avoiding differential attacks on fixed "optimal" tables. Thus

```
  ------------------------------mix------------------------------
  --------------mix-------------- --------------mix--------------
  ------mix------ ------mix------ ------mix------ ------mix------
  --mix-- --mix-- --mix-- --mix-- --mix-- --mix-- --mix-- --mix--
  mix mix mix mix mix mix mix mix mix mix mix mix mix mix mix mix
  S S S S S S S S S S S S S S S S S S S S S S S S S S S S S S S S
  mix mix mix mix mix mix mix mix mix mix mix mix mix mix mix mix
  --mix-- --mix-- --mix-- --mix-- --mix-- --mix-- --mix-- --mix--
  ------mix------ ------mix------ ------mix------ ------mix------
  --------------mix-------------- --------------mix--------------
  ------------------------------mix------------------------------
```

enciphers 256-bit blocks through 32 keyed 8-bit substitutions by using five levels of input keyed mixing transform and five levels of output keyed mixing transforms of varying size. Clearly, there are a plethora of alternate interconnection possibilities here. For example, the mixing rows could be permuted, different sizes of mixing combined in some rows, the mixing not arranged on 2^n boundaries, etc., etc. Since the mixing transforms are extremely fast, we would expect this 256-bit system to be much faster than 64-bit single-DES.

And,

```
  S S S S S S S S S S S S S S S S S S S S S S S S S S S S S S S S
  mix mix mix mix mix mix mix mix mix mix mix mix mix mix mix mix
  --mix-- --mix-- --mix-- --mix-- --mix-- --mix-- --mix-- --mix--
  ------mix------ ------mix------ ------mix------ ------mix------
  --------------mix-------------- --------------mix--------------
  ------------------------------mix------------------------------
  --------------mix-------------- --------------mix--------------
  ------mix------ ------mix------ ------mix------ ------mix------
  --mix-- --mix-- --mix-- --mix-- --mix-- --mix-- --mix-- --mix--
  mix mix mix mix mix mix mix mix mix mix mix mix mix mix mix mix
  S S S S S S S S S S S S S S S S S S S S S S S S S S S S S S S S
```

enciphers 256-bit blocks through 64 keyed 8-bit substitutions by using nine levels of mixing transforms of varying size. With the substitutions all keyed, we can probably avoid keying the mixing transforms. Again, there are a plethora of alternate interconnection possibilities.

Summary

Practical, high-speed, keyed, balanced, and size-preserving block mixing transforms are introduced for cryptographic service.

References

[1] Arko, R. 1961. Mechanical Signal Combiner. U.S. Patent 3,159,712.

[2] Beauchamp, K. 1984. Applications of Walsh and Related Functions. Academic Press.

[3] Brigham, E. 1974. The Fast Fourier Transform. Prentice-Hall.

[4] Geffe, P. 1973. How to protect data with ciphers that are really hard to break. Electronics. January 4. 99-101.

[5] Kohler, H. 1951. Combining Circuits. U.S. Patent 2,567,214.

[6] Massey, J., and R. Rueppel. 1989. Method of, and Apparatus for, Transforming a Digital Data Sequence into an Encoded Form. U.S. Patent 4,797,922.

[7] Rabin, M. 1980. Probabilistic Algorithms in Finite Fields. SIAM Journal on Computing. 9(2): 273-280.

[8] Ritter, T. 1990. Substitution Cipher with Pseudo-Random Shuffling: The Dynamic Substitution Combiner. Cryptologia. 14(4): 289-303.

[9] Ritter, T. 1991. The Efficient Generation of Cryptographic Confusion Sequences. Cryptologia. 15(2): 81-139.

[10] Siegenthaler, T. 1985. Decrypting a Class of Stream Ciphers Using Ciphertext Only. IEEE Transactions on Computers. C-34: 81-85.

[11] Smith, H. 1950. Combining Circuit. U.S. Patent 2,496,317.

[12] Vernam, G. 1919. Secret Signaling System. U.S. Patent 1,310,719.

Terry Ritter   ritter@cactus.org (alas, cactus.org dies March 18)
               ritter@io.com (perhaps temporarily)

Large Block DES

An on-line development project in cryptography.

The goal is a block cipher stronger than DES, and faster than Triple-DES. Ideally, it would also have a block size much larger than DES.

Contents

- Background
- NxM DES
- Ladder DES
- Large Block DES Newsletter
- Balanced Block Mixers
- Fenced DES
- Toward Axiomatic Fenced DES
- The Context of the Fenced DES Design
- Announcing Realized Prototypes
- Huge Block Size Discussion on sci.crypt

Background

The project began in the weakness of DES, and the horror of systems people contemplating the widespread use of Triple-DES. The goal was a block cipher stronger than DES and substantially faster than Triple-DES. A design was constructed and the NxM DES article prepared.

Alas, NxM DES was found to be weak *after* publication. This left the problem unsolved, and the failure (in itself, not all that unusual) embarrassingly exposed. Thus began the open development process.

A series of approaches and articles were prepared, culminating in the introduction of Balanced Block Mixers and the resulting Fenced DES cipher. While Fenced DES is a new approach, and so not easily accepted, it does appear to solve the problem.

NxM DES

The original article in the series, and the desirable Nx2 form is eventually shown weak.

- 1994-02-02 Terry Ritter: NxM DES
- 1994-02-02 Stewart Strait: "analogous in some sense to Playfair"
- 1994-02-03 John Taber responds to Stewart: "I'm not sure I understand all this. The DES itself is a simple substitution cipher on an "alphabet" consisting of 2^64 characters."
- 1994-02-07 Eli Biham responds to the original proposal through Bruce Schneier: "I claim that the cryptosystem described in your email is not stronger than a single DES, when it uses only two passes -- i.e., Mx2 DES.
- 1994-02-10 Terry Ritter: "before I open mouth wide and insert foot, I would like to be able to understand the attack that Biham describes."
- 1994-02-12 Terry Ritter: Nx2 DES Found Weak: the end of the line for Nx2 DES.

Isolated Double DES

An attempt to strengthen Double DES (which is known to be weak) to usability.

- 1994-02-17 Terry Ritter: Isolated Double DES
- 1994-02-20 Dave Sparks
- 1994-02-21 John Payson

Ladder DES

A higher-level "Feistel Cipher" using DES as the nonlinear function.

(NCSA Mosaic has been known to destroy the ASCII flow-diagrams in this article, because they accidentally include the HTML "comment" sequence. Netscape has no such problem, but if your browser does, it may help to view the document "source." I want to avoid changing anything within the documents.)

- 1994-02-22 Terry Ritter: Ladder DES
- 1994-02-23 Mark R: "Another, already somewhat widely used alternative is RSA Data Security's "DESX" "
- 1994-02-23 David Barber: "why don't we want to use IDEA to replace DES?"
- 1994-02-23 Mark Henderson responds to David: "For one, there is the issue of the patent on IDEA."
- 1994-02-23 Owen Lewis responds to David: "1. The massive investment already made in DES technology and equipment makes people reluctant to change. 2. For the US market, the NIH (Not Invented Here) syndrome."
- 1994-02-25 Terry Ritter responds to David: "I have trouble *believing* that IDEA is strong."
- 1994-02-25 Terry Ritter responds to Mark: " Where was this design published openly for comment? What were the comments?"
- 1994-03-01 Andrew Haley comments: "These schemes have been the subject of much research in recent years. The classic paper is [1], in which it is shown that three rounds in the structure above are sufficient to prove perfectness (polynomial time indistinguishability from a truly random permutation), provided that the random functions are themselves perfect."
- 1994-03-01 Mark Riordan: DESX definition: "DESX is an encryption algorithm that extends the famous DES (Data Encryption Standard) algorithm to a keysize of 120 bits."
- 1994-03-01 Terry Ritter: DESX overview: "Presumably the salient issue is that the outside XOR's are intended to protect the internal DES from exhaustive search."

Q: Am I the only one to find something odd about theoretical results which "prove" strength *provided* that the base functions are strong? If we had anything reasonable with provable strength, we wouldn't be going through this!

Large Block DES Newsletter

A summary of the never-ending project.

- 1994-03-02 Terry Ritter: Large-Block DES Newsletter

Balanced Block Mixers (originally called Block Mixing Transformations)

Simple, weak, fast and expandable mechanisms for mixing two input blocks into two output blocks. While many cryptographic mechanisms have do not have actual property proofs, this mechanism does.

Among other things, a Balanced Block Mixer construct provably *guarantees* to propagate a value change in one input block to both output blocks. This property can be used to produce *provable* overall diffusion or avalanche in a block cipher. While overall diffusion does not necessarily imply strength, overall diffusion is required in a good block cipher.

- 1994-03-13 Terry Ritter: Announcement (26K)
- 1994-03-14 Colin Plumb: "Basically, this operation is far too linear."
- 1994-03-15 Terry Ritter responds to Colin: " My intent in defining such constructs was to try and separate the concepts of "strength" from the "mixing" property. This means that we do not have to define what "strength" means in such a construct, but can instead rely on more classical forms, such as DES or substitution."
- 1994-03-15 Colin Plumb responds to Terry: "I see your idea now. Changing the widths like that will require a bit more thinking. The preservation of Z^B still applies, but the substitutions make it rather more complex."
- 1994-03-15 Terry Ritter responds to Colin: " All we need the mixings to do is to mix. Essentially, we want to end up with the effect of a bit change in any particular position being spread among the entire output (statistically), after a set of mixings. If this can be accomplished, we can use small, practical substitutions to make a large-block cipher."
- 1994-03-16 Colin Plumb responds to Terry: "Okay, time to comme clean: I get a bit annoyed when I see a fancily-formatted "report" that might fool some people when such glaring points as this are wide open. It seems enormously pretensious and makes me think poorly about the author. Sorry if it's unkind, but it's true that a part of my mind is saying "I wish Terry would quit posting his `clever' ideas until he learns thing one about cryptanalysis." "
- 1994-03-16 Terry Ritter responds to Colin: " Frankly, I think this says more about Colin Plumb than Terry Ritter. When Plumb publishes a practical cryptosystem which he can *prove* secure, then I'm sure the rest of us will take some notice. Until then, we all must learn to deal with -- and depend upon -- not only proposals, but actual fielded systems of unknown strength."
- 1994-03-16 Paul Crowley jumps in to aid Colin: "I'm with Colin on this one. Since sci.crypt isn't a refereed publication, don't try to make your articles look like "reports" when they have such clear defects."
- 1994-03-16 Prof. Dr. Klaus Pommerening: " I've learned some useful things from Terry Ritter's recent postings."
- 1994-03-16 Brian Olson jumps in to aid Colin: "Colin is one of sci.crypt's best posters. "The rest of us" already take notice, and couldn't help but notice that he fed you your technical lunch. Sorry Terry, but you deserved the slam."
- 1994-03-16 Colin Plumb responds to Terry: "I didn't mean to be insulting. I was feeling a little bit frustrated and trying to express it without being vicious. I read my words again and I still think I hedged them enough that it's not a personal attack. Am I wrong?"
- 1994-03-16 Terry Ritter: responds to Brian: " And exactly where is this "lunch"? All I see is one-liners and concerted giggling. I'm not sure anyone ever "deserves" a slam. Interesting that someone would think so, however."
- 1994-03-16 Terry Ritter responds to Paul: " I'm with me on this one. Since sci.crypt isn't a refereed publication, I'll format my reports however I like."
- 1994-03-17 Terry Ritter responds to Colin: " In my opinion, it was an attack, yes. It was also substantially misleading with respect to the quality of the article. It is appropriate to criticize the material. It is not appropriate to imply expertise so that your opinions alone will discredit a substantially-valid argument which you do not understand."
- 1994-03-17 Paul Crowley responds to Terry: "I have usually found free software as widely available as PGP to be more reliable and solid than commercial products..."
- 1994-03-18 John Kelsey: " Are you familiar with Merkle's Khufu and Wood's REDOC cryptosystems? Both of these use a scheme that could be used for cheap, fast mixing of blocks. The basic idea: To mix two 64-bit blocks, you first do some pre-processing, to generate a key table (Wood's term) of 256 64-bit entries."

I note in passing that the mixing that John proposes cannot be a *balanced* form of mixing. It also combines both "strength" and "mixing," and I'm not sure we know how to evaluate the result. Will strength make up for an unbalanced mixing? How much strength do we need? How much does it have?

As this thread trails off, nobody has been convinced that a weak mixing mechanism can have an advantage in a cryptographic design. Ironically, we now know that about three months earlier, related mixing mechanisms *were* included in a serious block cipher: In December 1993 the well known and respected cryptographer James Massey presented his design for the SAFER K-64 block at the fast software ciphering conference. SAFER K-64 uses "PHT's" or "Pseudo-Hadamard Transforms" which are similarly weak but are **not** *balanced* block mixers.

From a design point of view, one of the most important aspects of Balanced Block Mixing is the *provability* of change propagation. This is addressed in detail in the Fenced DES design.

Fenced DES

A wide-block structure having a single DES level which is isolated, both on input and output, by arrays of keyed substitutions. The input substitution results are mixed into four separate DES operations each using separate keys. The DES results are then mixed into the output fencing array.

The intent was to find a structure provably stronger than DES, which itself used DES as a component. The design goal was to find some sort of structure which used multiple DES operations to encipher a wider block. This provides much greater strength, with almost single-DES speed.

- 1994-04-29 Terry Ritter: Fenced DES (25K)

Toward Axiomatic Fenced DES

A preliminary attempt to formalize the concept of block mixing, and demonstrate that the mixing structure is inherently strong.

- 1994-05-26 Terry Ritter: Toward Axiomatic Fenced DES (43K)
- 1994-06-02 Bryan Olson: "Please don't call them theorems and proofs if they are not."
- 1994-06-02 Bryan Olson: "Oops."
- 1994-06-05 Terry Ritter responds to Bryan: " Please don't nit-pick new work to death. Few of these statements are actually false; in some cases the wording needs to be refined. Of course, some *are* defined so poorly that -- upon reflection -- they are probably not worth having.
  The concept of mathematical "proof" is subjective; we don't have a global automatic theorem-checker. We can't just toss this stuff in a machine and have it decide yes or no and where the weakness lies. If I had such a thing, my proofs would pass it.
  I obviously get complaints, but what, no complements? No comments on my proofs of the example Block Mixing Transforms? No complements for finding provable properties which can be used to expand the width of block ciphers as components? If you have been following this process, you know it has involved a serious amount of effort, trial-and-error, and public failure. Success was by no means assured. When it occurs it should be uplifting to all of us."
- 1994-06-05 Bryan Olson responds to Terry: "Thanks for reading my comments. You and I still disagree about the value of your results."
- 1994-06-06 Terry Ritter responds to Bryan: "I claim that Fenced DES is a design for producing efficient mechanisms (e.g., writing programs) which model a very large, invertible, key-selected substitution. I claim that such a substitution is the ideal model of any possible block cipher. I also claim that a mechanism based on the Fenced DES design can compute enough substitutions to have a large-enough keyspace to be cryptographically effective."
- 1994-06-07 Bryan Olson responds to Terry: "Unfortunately, your definitions and theorems don't establish this. If the identity function is an invertible substitution, then it is not the case that being an invertible substitution makes a function a good block cipher. You present no reason to believe that either fenced DES or your substitution boxes will have properties which are approximately average for invertible substitutions."
- 1994-06-09 Terry Ritter responds to Bryan: "It is standard practice to model a strong block cipher as a large invertible substitution. The keyed part of this is the selection of one substitution out of all possible substitutions. If we cared about the structure of a particular substitution, there would be better and worse keys, which would not be encouraging.
  This is a different situation than the small, fixed and known tables inside DES.
  Given this amount of confusion, I have obviously failed to provide a proper context for this exercise."

The Context of the Fenced DES Design

An overview of Fenced DES with some proofs and strength discussions.

- 1994-07-01 Terry Ritter: The Context of the Fenced DES Design (42K)

Announcing Realized Prototypes

- 1996-01-16 Terry Ritter: Five realized prototypes and comparative speed measures. (11K)

Huge Block Size Discussion

Discussion from the sci.crypt newsgroup.

- 1996-08-23 (150K): It is proposed that huge blocks would be appropriate for block cipher design.
- 1996-09-02 (65K): An attack on Fenced DES is handwaved, but turns out (after a great deal of arduous extraction of facts) to be *very* incomplete. **This is *not* a successful attack.**

Terry Ritter, his current address, and his top page.

Last updated: 1998-01-20

A Mixing Core for Block Cipher Cryptography

Cipher Huge Blocks with Guaranteed Mixing

A *Ciphers By Ritter* Page

Terry Ritter

A Mixing block cipher is distinguished by the exclusive use of Balanced Block Mixing (BBM) structures for block diffusion.

In dramatic contrast to conventional designs, here *all* diffusion occurs in simple *linear* BBM operations **which are not intended to provide strength!** By itself, linear mixing has no strength at all, but it does set the stage for keyed substitution tables which *do* have strength. The BBM's provide a *guaranteed*, exactly balanced and reversible mixing which, in a successful design, prevents individual tables from being distinguished and solved.

Here we present the ciphering ***core,*** the architecture of tables and mixing which performs actual ciphering. The tables are initialized or "keyed" by a separate process which shuffles each table independently, twice.

Mixing Cipher Attributes

- **SPEED** -- about 1,005,000 bytes per second in 16-byte blocks and about 980,000 bytes per second in 64-byte blocks (on a 100 MHz "686," under Win95, with the Borland 32-bit Delphi 2 compiler). The typical keying overhead is around 64 milliseconds for 64 tables.
- **STRENGTH** -- greater than 128 bits for a block size of at least 64 bits and at least 24 keyed tables.
- **BLOCK SIZE** -- dynamically selectable at ciphering time in power-of-2 steps. A single program handles "legacy" 64-bit blocks, "modern" 128-bit blocks, "independent" 64-byte blocks and 512-byte disk blocks without change.
- **KEY SIZE** -- arbitrary, independent of block size.

Contents

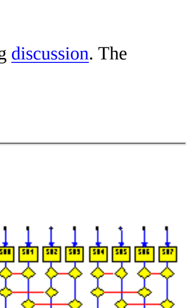
- Mixing Cipher Attributes
- Background
- Components
- Scalability
- The BBM
- Using The BBM
- The BBM in FFT Patterns
- A 64-Bit Mixing Core
- Deciphering
- Large Blocks
- Primary Keying
- Strength
- Dynamic Keying
- Authentication
- Strength Arguments
- Summary of Advantages
- Also See:

Background

The first known publication of the concept of Balanced Block Mixing and its use to build Mixing ciphers was the 1994 Mar 13 sci.crypt announcement by Ritter and the resulting discussion. The general lack of understanding in the responses shows that the concept was new to those who replied.

Mixing ciphers are protected by patent.

Components



In a Mixing cipher core we have exactly two component types:

1. keyed, invertible substitution tables, and
2. Balanced Block Mixers (BBM's).

```
In the figure, each box represents a table, and each connected pair of diamonds represents one BBM. Plaintext data
bytes enter at the top, are substituted, mixed, substituted, mixed and substituted into ciphertext.
```

The essence of the design is an inability to externally isolate or distinguish one table from another. This inability is directly related to the exact balance in the high-quality BBM mixing, as opposed to an exponential increase in complexity from repeated low-quality mixing with confusion.

Tables are widely used in many designs, and even keyed tables are not that unusual. But in this design *all* of the inter-element diffusion occurs in distinct mixing layers composed of *linear* Balanced Block Mixing (BBM) operations, and this *is* new.

This separation between mixing (diffusion) and keyed substitution (confusion) is beneficial in that the two operations can be seen, understood, and measured separately. In conventional designs, confusion and diffusion are often combined, which makes these ciphers difficult to understand *or* measure. And conventional ciphers do not scale down to testable size.

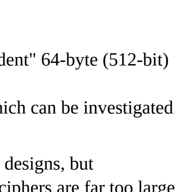
Scalability

A Mixing cipher is scalable in a way that few designs are:

- First, the block size is dynamically selectable in power-of-2 steps. A single unchanged program can handle "legacy" 64-bit blocks, "modern" 128-bit blocks, and "independent" 64-byte (512-bit) blocks.
- Next, we can even scale down the size of the tables and the mixing elements. This means that the same design can deliver either a full-scale real cipher, or a tiny model which can be investigated experimentally for weakness.

This ability to scale a single design between large real ciphers and small testable models almost cannot be overemphasized. Cryptography **does** have better-known structures and designs, but cryptography simply **does not** have a proven methodology which -- if only followed -- will guarantee a strong design. This means that testing is exceedingly important, and real ciphers are far too large for many types of test.

The BBM

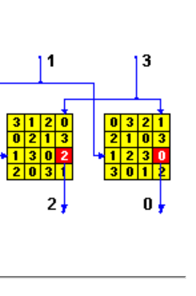


A Balanced Block Mixer can be considered an orthogonal pair of Latin squares. The two inputs select "row" and "column" and return the value from the selected element of each square. This "mixes" or combines the two input values into the two output values with some useful properties for this design:

1. A change to either input will change both outputs.
2. No input change can keep both outputs unchanged.
3. Any possible value (value balance) from either output can be achieved by changing either input (structural balance), regardless of the value on the other input (independence).

Even a single bit change in one mixing input will produce some change in both mixing outputs. And if each output is substituted through a keyed table, each table will produce a random result value. So even a single bit change on one input will produce a new random result on both outputs.

Using The BBM

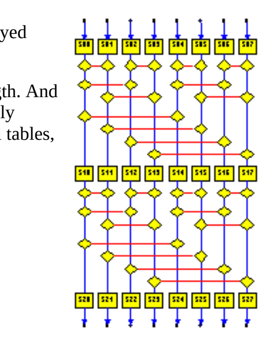


One way to use BBM structures is to create full-blown Latin squares: The typical "8-bit" mixing would thus require two 64K byte orthogonal tables. Now, 128K of keyed tables *will* combine strength with guaranteed mixing, but large keyed Latin square tables are harder to create than we might like.

The aim of the Mixing construction is different: Here we use simple, linear, computational "Latin squares" for mixing, and then we add smaller keyed tables for strength. And there is plenty of strength around: we can easily realize many thousands of keying bits even with fairly small tables (a shuffled "8-bit" table of 256 byte elements ideally represents 1648 bits of keying). So, in a sense, strength is not the problem; instead, the problem is in preventing The Opponent from isolating and analyzing individual tables, and this is a *mixing* issue.

The BBM in FFT Patterns

In Mixing ciphers, we typically use small "8-bit" BBM's. This gives us a reasonable 8-bit table size, and the linear BBM mixing is easy and fast. But this means that somehow we will have to mix at least 8 tables, not just 2.



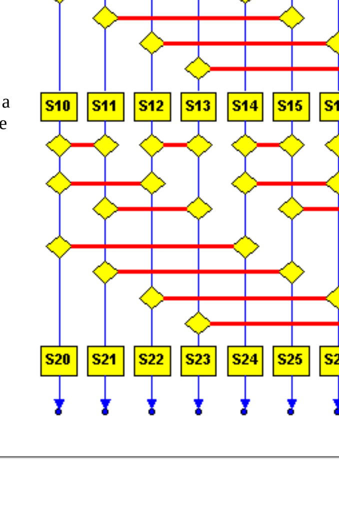
```
In the figure, we see 3 "sub-layers" of BBM mixing. Each pair of values are mixed, and the results handed on
to the next mixing sub-layer. By tracing the data paths, we can see that each and every input value from the
top is represented exactly once in each and every output value at the bottom.
```

We can expand BBM's by mixing in FFT-like or FWT-like patterns (see An Efficient FFT-Style Mixing Pattern, on these pages). The result is essentially a single BBM with n-inputs, n-outputs, and properties analogous to the 2-input 2-output component:

1. A change to any input will change all outputs.
2. No input change can keep all outputs unchanged.
3. Any possible value (value balance) from any output can be achieved by changing any input (structural balance), regardless of the value on the other inputs (independence).

Even though a linear BBM structure is inherently weak, the same can be said of exclusive-OR, which is used in presumably strong ciphers all the time. The balance and independence properties created by a wide BBM force an even statistical distribution across all values and all outputs. This mixing seems ideal for coupling keyed tables into a large effective block.

A 64-Bit Mixing Core



The diagram shows the Mixing cipher core for an 8-byte (64-bit) block. Note that this structure is dynamically scalable to handle blocks of arbitrary power-of-2 size at ciphering time, simply by doing more mixing sublayer computations.

The thin vertical blue lines represent a value moving down through time. Each yellow diamond represents *half* of a Balanced Block Mixing and a value change. Two diamonds connected by a thick horizontal red line represent a *complete* byte-wide BBM operation.

Each block (e.g., S00, S15, etc.) represents an individually keyed table substitution. The actual table used in each position will be selected from an array of such tables. Each table is a keyed invertible substitution.

At the 8-byte width, each byte is processed through a BBM three times before we have a full mixing. Each doubling of the block width implies another sublayer of BBM processing, so a 64-byte (512-bit) block would need 6 sublayers, a 512-byte (4096-bit) block would need 9 sublayers, and a 4096-byte (32,768-bit) block would need 12 sublayers.

An ideal hardware realization would have a separate on-chip table for each position in three confusion layers across a 64-byte block. This would imply a particular keyed table for each of 192 table positions.

In software, we can have a smaller number of tables and re-use them sequentially wherever a table is required. Note that it is this ability to re-use a fixed number of tables which allows us to handle arbitrarily huge blocks. If we have exactly 192 tables, the software realization can be compatible with the ideal hardware.

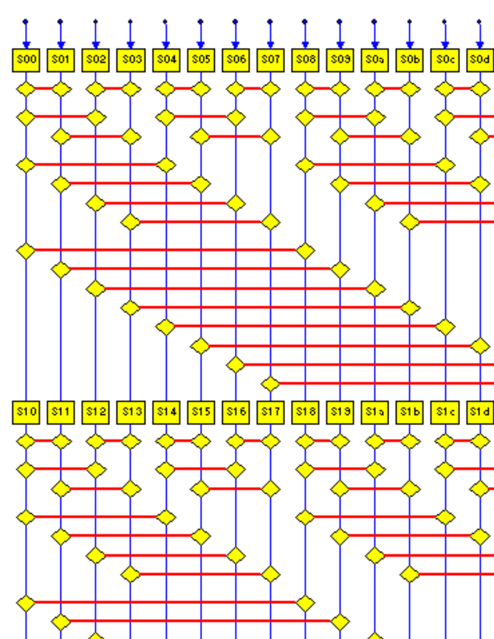
Deciphering

Since we use invertible substitution tables, we can "undo" any table translation by sending the resulting value through the corresponding inverse table.

The inverse of the BBM component used here is just another occurrence of that same BBM component. Similarly, the FFT-like full mixing layer is just another occurrence of that layer.

This means that *exactly the same ciphering routine (or hardware)* can be used for both enciphering and deciphering. For deciphering we use inverse tables and take the initial values (IV's) in reverse layer order. (The IV's are the starting table values for each layer.) Both of these conditions are easily handled during keying.

LargeBlocks



DES has a fixed block size and we have somehow managed thus far, so it may seem like overkill for a cipher to have multiple block sizes. But there is almost no cost for this, and it must be admitted that having blocks of various size sometimes can provide a better fit to the overall system than any single fixed block size. But perhaps the biggest benefit comes from the ability to cipher in very large blocks.

Large Blocks are Strong

If plaintext really does contain uniqueness at a rate of only about one bit per character, a legacy 64-bit block covers only about eight bits of uniqueness. This is the situation encountered in the classic codebook attack. This sort of attack is *not* avoided by having a larger keyspace, but *can* be avoided by using a wide, unbiased selection of plaintext blocks. Normally this is done by using a chained operating mode to randomize the plaintext. But increasing the block size to 64 bytes or more can collect enough uniqueness in the plaintext block so that randomization can be avoided.

Large Blocks Can Be Independent

By increasing the block size to 64 bytes or more we may be able to operate in "electronic code book" (ECB) mode instead of "cipher block chain" (CBC) mode. This means that we may not need to develop, send or store an initial value (IV), which would otherwise expand the ciphertext. And it also means that blocks can be both enciphered and deciphered independently, even if they are received out-of-sequence, as may happen in packet-switching transmission.

Large Blocks Have Room for Dynamic Keying and Authentication

In conventional block cipher designs, the block size is so small that we can scarcely consider displacing some data with other information. But when we have a large block, there is room for other information, at a relatively small overhead. Typical applications include Dynamic Keying and Authentication.

Primary Keying

Primary keying generally consists of shuffling each substitution table with a keyed cryptographic random number generator (RNG), twice. Primary keying can use fairly conventional technology, and is largely independent of the ciphering core itself. One example of a keying system is described in A Keyed Shuffling System on these pages.

Primary keying takes about 1 msec per table on a 100 MHz processor, which is fast enough on a human scale. Primary keying does take time, because it is specifically intended to perform as much computation as possible -- once -- instead of having computation repeated with every block at ciphering-time.

Strength

We assign an 8 bit strength to each table. Although an arbitrary table permutation contains 1684 bits of independence, we reason that if The Opponents can find (that is, absolutely identify and confirm) one value, they can probably find another. Since each table value is 8 bits, we assume a table strength of just 8 bits.

With this sort of three confusion layer structure, we believe that only two of the layers contribute independent strength. Therefore, in a small 8-byte block, we expect to see 8 * 16 = 128 bits of strength in 24 separate tables.

Dynamic Keying

True zero-latency dynamic keying is available by placing keying values in each data block along with data. This will of course expand the ciphertext by the size of the keying field, but even a 64-bit dynamic keying field is only about 12.5 percent of a 64-byte block. This sort of keying can be used in any true (that is, avalanching or data diffusing) block cipher with room for extra data.

Authentication

Strong block-by-block authentication is available similar to dynamic keying. Authentication values are placed into each data block along with data. Potentially, this can avoid a higher-level scan across the data with a cryptographic hash function. The exact same field can provide both authentication *and* dynamic keying.

Strength Arguments

Here we present various attacks and comment on their likelihood of success on this particular cipher. Recall that attacks are not *algorithms*, but instead just general approaches which must be reinvented for every new type of cipher.

Exhaustive Search (Brute Force on the Key)

> Try each possible key until the message deciphers properly. Try most-likely keys first.

A keyspace of at least 120 bits should be sufficient to prevent exhaustive search in the foreseeable future. The keying system for the Mixing core has a keyspace substantially beyond this value, mainly because this produces a convenient design.

No cipher can do very much about key search attacks if there are only a relatively small number of possible keys, and if some keys are vastly more probable than others. It is the responsibility of the larger system to prevent this.

Chosen Key

> Try various keys on known plaintext and compare the resulting ciphertext to the actual ciphertext, to try and build the correct key value.

If a user has the ability to generate specific keys which are used by the Mixing core on data, it is quite likely that the external cipher system has already failed. However, even in this situation, key selection seems unlikely to help The Opponents. Sure, they can force particular table values by manipulating the key, but they can do that *without* going through the keying process. The Opponent's main problem in attacking a Mixing cipher is that the mixing appears to couple the various tables together so that a single table cannot be isolated and worked on separately.

Ciphertext Only

> The Opponent accumulates a mass of ciphertext material and tries to find relationships within the data.

This is a general class of various specialized attacks which all use only the exposed ciphertext as opposed to particular knowledge of the plaintext or access to the ciphering system itself.

Ciphertext-Only Codebook

> Collect as many ciphertexts as possible and try to understand their contents through usage and relationships; then, when a ciphertext occurs, look it up. This treats the block cipher like a *code*, and is the classic approach to code-breaking.

Just as some letters are more frequently used than others, words and phrases also have usage frequencies, as do blocks which contain plaintext. If the cipher block size is small (under 64 bytes), and if the plaintext is not randomized, and if dynamic keying is not used, and if the ciphering key is not changed frequently, it may be possible to build a codebook of block values with their intended meanings.

Codebook attacks of any sort are ideally prevented by having a large number of block values, which implies a large block size. Once the block size is at least, say, 64 bytes, we expect the amount of uniqueness in each block to exceed anyone's ability to collect and form a codebook.

Since the complexity of any sort of a codebook attack is related to block size only, doing "triple" anything will not affect increase this complexity. In particular, this means that Triple DES is no stronger that DES itself, under this sort of attack, which is based on block size and not transformation complexity.

Known Plaintext

> Somehow "obtain" both the plaintext and the corresponding ciphertext for some large number of encipherings under one key.

First, since the Mixing core described here has an internal state typically 512 times as large as a 64-byte data block, we know that a single plaintext and ciphertext pair simply do not contain sufficient information to reveal the full internal state. Note that a *single* known plaintext and ciphertext pair probably *would* identify a DES key.

Larger amounts of known plaintext and ciphertext will of course surpass the required information, but the question is how The Opponent might use it. The problem is a supposed inability to distinguish one table from another and so work on one table at at time.

Known-Plaintext Codebook

> Collect as many ciphertexts and associated plaintext blocks as possible; then, when a ciphertext occurs, look it up.

Small block ciphers prevent codebook attacks by randomizing the plaintext (often with Cipher Block Chaining) so that the plaintext block values are distributed evenly across all possible block values. But not all block ciphers are always applied properly.

Codebook attacks are ideally prevented by having a large number of block values, which implies a large block size. To prevent this attack for the future, we need a block size of at least 128 bits, and even then *still* require the plaintext to be randomized. If we wish to avoid randomizing with CBC, we need a block which is large enough so the uniqueness it does contain assures that there will be too many different blocks to catalog. A reasonable power-of-2 minimum size to avoid randomization would be at least 64 bytes.

Chosen Plaintext

> Without knowing the key, arrange to cipher data at will and capture the associated ciphertext. Dynamically modify the data to reveal the key, or keyed values in the cipher.

The point here is not to decipher the associated ciphertext because The Opponent is *producing* the original plaintext. If The Opponents have chosen plaintext capabilities, they can probably also submit arbitrary ciphertext blocks for deciphering.

The weakness to be exploited here depends upon the ciphering system beyond the core cipher *per se*. If the key values change with each message, and the ciphering keys are not under the control of the user (if the system uses message keys), there simply *is* no fixed internal state to be exposed.

If the primary key remains the same for all messages, then there will be some fixed state to try and ferret out. But if the ciphering system uses dynamic keying fields (with values again not under the control of the user), there can be no completely-known Chosen Plaintext blocks for use in analysis.

If the ciphering core is used raw, without primary re-keying and also without dynamic keying, the question arises as to whether there exist any statistical relationships which can be exploited better by Chosen Plaintext than by Known Plaintext. None are known.

Chosen-Plaintext Codebook

> Create as many ciphertexts and associated plaintext blocks as possible; then, when a ciphertext occurs, look it up.

This is much like the previous codebook attacks, now with the ability to fill the codebook at will and at electronic speeds. Again, the ability to do this depends upon the cipher not having dynamic keying fields. How difficult this will be depends upon the size of the block.

Meet-in-the-Middle

> With a multi-layered structure, given known-or defined-plaintext, search the top keyspace to find every possible result, and search the bottom keyspace to find every possible value.

With a two-level construct and a small block size, matches can be verified with a few subsequent known-plaintext and ciphertext pairs. Of course, three and more-level constructs can always be partitioned into two sections so a meet-in-the-middle attack can always be applied. It just may be pretty complex.

The Mixing core avoids meet-in-the-middle attacks by using a three-level construction, in which each layer has a huge amount of keyed state or "keyspace."

Key Bit Bias

> Through extensive ciphering of fixed plaintext data under a variety of different keys, it may sometimes be possible to associate key bits with the statistical value of some ciphertext bits. This knowledge will break the cipher quickly.

This is a rather unlikely circumstance, albeit one with absolutely deadly results.

Differential Cryptanalysis

> Exploit known properties of particular known substitution tables to effectively reduce the number of "rounds" in an iterated block cipher.

The original form of Differential Cryptanalysis mainly applies to iterated block ciphers with known tables, neither of which are present here. However, the general concept is more generally applicable.

Since each input byte is mixed into every mid-level substitution operation, and these are mixed again into output substitutions, it is hard to see how Differential Cryptanalysis could be used.

Summary of Advantages

Strength

- The strength advantage of large blocks, which can hold huge amounts of plaintext uncertainty.
- Large blocks have room for dynamic keying information, which supports zero-latency block-by-block keying.
- Large blocks have room for authentication information, which can avoid the need for a slow higher-level authentication pass over the data
- The strength advantage of massive, keyed, nonlinear, hidden internal state, which generally means that an attacker must somehow *expose* that state.
- The strength advantage of a fundamentally *scalable* design which supports tiny true models that can be exhaustively tested.

Flexibility

- The flexibility advantage of dynamically selectable block size. This directly supports "legacy" 64-bit blocks, "modern" 128-bit blocks, "independent" 64-byte blocks, and 512-byte disk sectors in *the exact same unchanged program*.
- Large blocks can eliminate the need for plaintext randomization, chaining, and the use of ciphertext-expanding IV's.
- Large blocks support the independent ciphering of blocks, and random-access ciphering.
- The dynamically selectable block size can limit ciphertext expansion by stepping down through smaller blocks at the end of a message.
- The flexibility advantage of key processing as a part of the cipher. This directly supports *both* textual *and* random binary keys of arbitrary length.

Hardware

- The hardware advantage of a constant ciphering rate *per block*, independent of the size of the block. Each pair of bytes across the block in each mixing sublayer can be simultaneously processed in parallel hardware, and each sub-layer can be pipelined, for gigabyte ciphering rates in present technology.
- The hardware advantage of a structurally simple design based on exactly two very dense chip structures (tables and BBM components). Ideal for tight chip layout.

Also see:

- The Large Block DES Open Development: our search for a new "DES" architecture started over four years ago.
- The original BBM article
- The start of a theoretical basis.
- Fenced DES
- The current BBM article
- The keying system
- The FFT mixing pattern
- Mixing on the top page

Terry Ritter, his current address, and his top page.

Last updated: 1998-03-21

A Keyed Shuffling System for Block Cipher Cryptography

Text or Binary Keys of Arbitrary Size
Create Keyed Invertible Substitutions

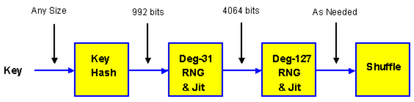
A *Ciphers By Ritter* Page

Terry Ritter

Many modern ciphers are based on keyed invertible substitution tables. Each such table contains exactly the same values as any other, but in a different order. In this environment, *keying* consists of constructing a *particular* arrangement of the data in each table. Ideally, every possible arrangement or *permutation* would be equally likely.

Separating the keying system from the actual ciphering allows the ciphering *core* to be as simple and fast as possible. This separation also allows each system to be analyzed largely independently.

Keying System Overview



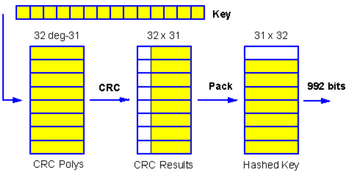
The goal of the keying system is to construct arbitrary invertible tables under the control of a key. In a sense, the arrangement of values in each table is the "real" key for the cipher. The keying system simply *processes* the external key into the larger internal form required by the design.

This key processing can be accomplished by using the external key to set the initial state for a random number generator (RNG), then using the RNG to shuffle the tables. Here we also use an intermediate RNG which expands the data and provides nonlinearity in the original state of the keying RNG, which helps protect against attack.

It is first necessary to convert an external key of arbitrary content and length into the fixed amount of data suitable for initializing a serious RNG. The general name for this process is *hashing*. By including hashing as part of the cipher, we can directly handle text keys, binary keys, voice keys, picture keys, file keys, or keys based on any information whatsoever, and of arbitrary size. And if we want a weaker cipher, we can just agree to use short external keys.

The hash result is used as the state or "seed" for the first RNG. The pseudo-random sequence from the first RNG is filtered to make it nonlinear and stronger, and the resulting sequence is used to construct the state for the second RNG. The second RNG produces a pseudo-random sequence which is also nonlinearly filtered. The resulting sequence is used to shuffle the tables.

The Hashing Subsystem



We wish to use a first RNG having 31 elements of 32 bits each, so it is convenient to develop a hash result of 992 bits from the variable-size key. For a hash of this size, we use conventional cyclic redundancy check (CRC) technology with a set of 32 different degree-31 mod-2 primitive polynomials.

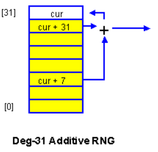
Unlike *ad hoc* cryptographic hash functions, CRC is well understood and has a strong mathematical basis. (For an introduction, see Ramabadran and Gaitonde [5]; CRC is also mentioned in Sect 5.3 of Ritter [6].) Although CRC is a linear function, and thus contributes no added strength, it is here used only for key *processing*, so no added strength is needed or wanted.

Each CRC scan across the key uses a particular degree-31 polynomial, and produces a 31-bit result in a 32-bit element. So by performing 32 different CRC's using each of 32 different polynomials, we accumulate 32 results, each of which has a most-significant-bit (msb) of zero. We then delete the msb's and pack the 32 elements of 31 valid bits into 31 elements of 32 bits each. This also causes the 32 CRC results to each start at a different bit-position in the packed array.

In our current design, each CRC hashing polynomial has an approximate balance in ones and zeros. These polynomials had to be constructed, because polynomials meeting the balance requirement were not available in the literature. The general ideas behind finding irreducible and primitive mod-2 polynomials are discussed in Ritter [6], Sect 6.5. A CRC only needs irreducibles, but at degree 31 these are all primitive anyway.

CRC hashing gives us a way to handle keys of arbitrary content and size, in a fixed, and relatively easily analyzed RNG system. The key can be a text string or a random binary value of arbitrary length. Keys of any reasonable size are thus directly supported in a fixed-size hash machine.

The First RNG



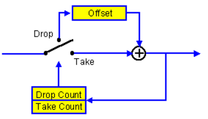
Deg-31 Additive RNG

We wish to use some form of RNG to key the ciphering tables. Our published survey of ciphering RNG's [6] (Ritter, in particular Sect 4.10), revealed the Additive RNG as a very attractive choice. An Additive RNG has a large internal state, a huge proven cycle length, and especially fast operation. It is easily customized by using an appropriate primitive mod-2 feedback polynomial. And -- unlike many cryptographic RNG designs -- an Additive RNG supports *almost* arbitrary initialization.

The Additive RNG is described in Knuth II [2: 26-31]. Also see Marsaglia [3] for comments, and Marsaglia and Tsay [4] for a sequence length proof.

We wish to use a deg-31 Additive RNG with 32-bit elements, which thus requires a 992-bit initialization value. We obtain this by hashing the external key into a 31-element array of 32-bit values. (We also scan this array looking for '1'-bits in the least-significant-bit (lsb) of each element; if none are found, the lsb of element [0] is made a '1'.) For best acceptance, the mod-2 primitive trinomial is taken from Zierler and Brillhart [7: 542].

The Jitterizer Nonlinear Filter



Jitterizer

The Additive RNG is a linear system; the sequence of values it produces is inherently linearly related. This is a potential weakness, which we avoid in several ways. One way is to use a nonlinear filter which we call a "Jitterizer." (The concept of a Jitterizer was introduced in Sect 5.5 of Ritter [6].)

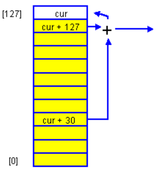
The approach taken here is to "drop" or delete a pseudo-random number of elements from the sequence, then "take" or pass-along another pseudo-random number of elements. Each of the elements in a "take" group is also exclusive-ORed with one of the dropped values.

When called in the "drop" phase, the Jitterizer calls the RNG extra times until the first element of the next "take" phase is reached. This value is offset and returned. When called in the "take" phase, the Jitterizer just calls the RNG once, offsets the value and returns it.

Our current Jitterizer design is constructed to drop 2..5 elements (avg. 3.5), then take 1..16 elements (avg. 8.5). This will drop out almost 30 percent of a long sequence. Each *last* dropped value becomes a new offset, so the 32-bit offsets change about every 8 or 9 returned values. Each *first* dropped value is offset, XORed down to a byte, and used to set take and drop counts.

The term "jitterizer" is used as a counterpart to the "synchronizer" mechanisms commonly used in digital design. An oscilloscope display which is not synchronized is said to "jitter." The purpose of a jitterizer is to create a non-synchronized stream.

The Second RNG



Deg-127 Additive RNG

We wish to use a deg-127 Additive RNG to control the actual shuffling. Such an RNG has an internal state of 4064 bits, which we originally develop by running the deg-31 RNG until we get 127 Jitterized 32-bit elements. We also scan the resulting state to assure that at least one '1' occurs in the lsb position, and increment element [0] otherwise.

The mod-2 primitive trinomial is again taken from Zierler and Brillhart [7: 542]. The sequence from the second RNG is also jitterized and used to control the shuffling.

The Shuffling Subsystem

First we initialize the tables in a counting pattern. Then we shuffle the tables. Twice.

The shuffle routine is the standard Durstenfeld [1] shuffle described in Knuth II [2: 139 §3.4.2.P] and mentioned in Sect 6.7 of Ritter [6].

Suppose we have "byte wide" substitution table, an array of 256 byte elements, from index [0] through [255]: To make a random arrangement, we first select one of the entries by creating a random value on the range 0..255. We then take the value in that element and place it into some known position; it is convenient to place the value at the far end of the array. This leaves us with a empty place in the array, and an extra value which came from the far end. We might shift the second part of the table down to fill the hole and make room for the displaced value, but it is easier to simply put the displaced value in the empty position. This amounts to exchanging the values in the randomly selected position and the end of the array.

With one value positioned, we have 255 values remaining, in positions [0] through [254]. So we use a random value in the range 0..254 to select a position, and exchange the selected value with the value in position [254]. Sometimes these are the same, which is fine. We similarly fill positions [253], [252], etc., down to position [1], which also defines the remaining value in position [0].

Perhaps the hardest part of this process is the development of unbiased random values with the correct range at each step. The Additive RNG's will generate random binary 32-bit values. It is certainly possible to convert the random values to reals on the range 0..1 and then multiply by the current position. Or we could do some sort of scaled integer multiply.

Another approach is to simply get a random value, and if the result is in our desired range we accept it, otherwise we reject it and try again. Now, obviously, the likelihood of getting a value in 0..253 from a 32-bit random value is pretty low, so we first convert the 32-bit value into an 8-bit value by exclusive-ORing high and low halves, twice. Similarly, the likelihood of getting a value in 0..5 from an 8-bit random value is also pretty low, so we arrange to *mask* the random value in power-of-2 steps. To look for a value in 0..5, we mask the 8-bit random to 3 bits, giving a range of 0..7, so in this case we expect to find a value in our range 6 times out of 8. This approach has the advantage of clearly giving an even probability in each desired range, something which is often not as clear when using reals and scaled multiplies. The approach also skips RNG values occasionally (about 1 time in 4, on average), which adds further uncertainty to the sequence position of each RNG value used.

Strength Arguments

Since RNG is mainly used to key the tables (it does also develop the few initial values needed by the core), an attack on the RNG can be made only *after* an Opponent has somehow developed some amount of table content. Note that a single "byte-wide" substitution table (an array of 256 byte values) represents 1648 bits of information.

The external keys are assumed to be unknown, since, if we do not make this assumption, The Opponent can simply enter the correct key and decipher the data directly. These are secret keys and we assume that they have been distributed by some secure process.

We do not particularly care if the mechanism protects the value of the external key, since the state of the ciphering core alone is sufficient to decipher data. An Opponent who can somehow develop the core state has no *need* for the original key.

We **do** seek to defend against any ability to use some amount of known state to develop knowledge of additional state. We seek to defend against using the recovered state of one table to develop the content of other tables. We thus assume some amount of known state, and seek to prevent that knowledge from being extended to other entries or tables.

There are basically four levels of resistance in the keying system:

1. **Large Keyspace:** A large keyspace defends against a brute force attack on the keys: We wish to have so many keys that it is impractical to try every key. The current thinking is that a 120-bit keyspace should be sufficient, since it seems impossible to try 2^{120} keys. The 992-bit keyspace in our current design is largely the result of design convenience. Having brute-force strength beyond 120 bits is largely of *theoretical* advantage, since a 120-bit keyspace is already impossible to search.
2. **Double Shuffling:** Each table is shuffled *twice*. At least 3296 bits of information are used to construct each table, even though a particular table arrangement represents only 1648 bits. Thus, we expect that double shuffling will give us 2^{1648} different ways to construct each and every possible table. This is an unsearchable quantity. Consequently, even an Opponent who has recovered the complete contents of a table still does not know what sequence was used to construct it. And knowing the sequence would seem to be a requirement for attacking the RNG.
3. **Nonlinear Filtering**: The Opponent then encounters the Jitterizer on the second RNG. Not only does this make the sequence nonlinear, with a different offset every 8 or 9 values, but about 30 percent of the sequence is simply *gone*. The Additive RNG is a form of linear feedback shift register (LFSR) and can be attacked by algorithmic means. But those attacks require that we know particular bit values and their relative positions. This information is what the Jitterizer destroys.
4. **Large State Source:** Finally, the Opponent encounters an RNG with 4064 bits of nonlinearized state. Since any particular table represents only 1648 bits of information, it is clear that the equivalent of at least two complete tables must be solved before reconstructing the second RNG. But if The Opponent can precisely resolve over 512 8-bit values in their correct table positions, the cipher is almost certainly broken anyway, and there is no need for an attack on the RNG.

These arguments form the basis for an assumption of sufficient strength in the keying system.

References

1. Durstenfeld, R. 1964. Algorithm 235, Random Permutation, Procedure SHUFFLE. *Communications of the ACM.* 7: 420.

2. Knuth, D. 1981. *The Art of Computer Programming, Vol. 2, Seminumerical Algorithms.* 2nd ed. Addison-Wesley: Reading, Massachusetts.

3. Marsaglia, G. 1984. A current view of random number generators. *Proceedings of the Sixteenth Symposium on the Interface Between Computer Science and Statistics.* 3-10.

4. Marsaglia, G. and L. Tsay. 1985. Matrices and the Structure of Random Number Sequences. *Linear Algebra and its Applications.* 67: 147-156.

5. Ramabadran, T. and S. Gaitonde. 1988. A Tutorial on CRC Computations. *IEEE Micro.* August: 62-75.

6. Ritter, T. 1991. The Efficient Generation of Cryptographic Confusion Sequences. *Cryptologia.* 15(2): 81-139.

7. Zierler, N. and J. Brillhart. 1968. On Primitive Trinomials (Mod 2). *Information and Control.* 13: 541-554.

Terry Ritter, his current address, and his top page.

Last updated: 1997-08-22

Efficient FFT-Style Mixing for Block Cipher Cryptography

Mixing Large Blocks with Byte-Size BBM's

A *Ciphers By Ritter* Page

Terry Ritter

For those who have not previously encountered FFT-style mixing, we present description, figures, and routines in both Pascal and C for a straightforward and efficient mixing pattern. We propose this as the *standard* pattern for large-block mixing. This pattern can be dynamically varied to cover arbitrary power-of-2 size blocks at ciphering time.

Source code is presented for bona fide informational or experimental purposes only: The mixing mechanism is protected by patent.

Mixing Two Elements

We start out by picturing the mixing of two elements by a single Balanced Block Mixing or BBM. In many cases this will take two byte values into two different byte values.

In the figure we have two vertical blue lines which just represent two element values flowing down through time. When a line is bare, the previous value is unchanged, but a yellow diamond indicates a value change.

Here there are two yellow diamonds, one on each line at the same time, connected by a broad red line between them. The yellow diamonds indicate a change in each value at a particular point in time and the red line connects the two parts of a single Balanced Block Mixing *pair*.

If we are mixing just two elements, there really is no wide range of possibilities -- we just mix one with the other. But when we have we have more elements, we have more possibilities.

Mixing Two Elements

Mixing Four Elements

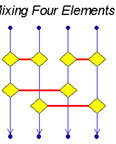
Next we consider mixing *four* elements, again using the fundamental two-element BBM mixing. Now we need *four* mixing operations, where before we only needed one. Although we might wish for a larger fundamental mixing operation which we would apply only once, we are in fact developing the internal process of making just such an operation.

If we have four elements to be mixed as *pairs*, one way to start is by mixing the first two elements and also mixing the second two elements. In this way, each of the first two element values is made a balanced result of both of the first two inputs, and each of the second two elements is made a balanced result of the second two inputs. In hardware, these mixing operations can occur simultaneously, and constitute what I call a mixing *sublevel*.

The point of the mixing is to end up with values such that each result value has been determined by each input value. We can do this by mixing an element from the first pair with an element from the second pair, and similarly mixing the remaining element from each pair, in a second *sublevel*.

Here, the selection of what to mix with what can be made in exactly two different ways, and if we just want *some* balanced mixing, there is no particular reason to choose one over the other. But some mixing patterns probably are a little more transparent than others, and some may be a little easier to automate.

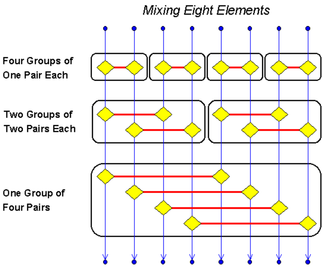
Mixing Four Elements



Mixing Eight Elements

Next we consider mixing *eight* elements, again using the fundamental two-element BBM mixing. Now we need 3 sublevels of 4 mixing pairs each. Again note that, in hardware, the 4 mixings in any sublevel can proceed simultaneously.

Mixing Eight Elements



To develop a general algorithmic description of this process it is reasonable to introduce the concept of a *group* of pairs. In the first mixing sublevel, we have only one pair in each of four groups. But in the second sublevel, we have two pairs, and two groups. And in the third sublevel, we have four pairs, and only one group.

We distinguish *pairs* and *groups* because they involve different forms of indexing through the array of data. Pairs start out separated by a value which is the number of pairs in each group at that sublevel. In the first sublevel, there is always one pair per group, and the first pair in the first group is always [0,1]. Then we step each element by one and, if we have done all pairs for that sublevel (and we always will have, in the first sublevel), we step to the next group.

Once we have done a pair, we need to step across the width of the data affected by the pair to the next undone element. But if we have just done the previous group (and stepped beyond it), all we have to do is increment each element by the number of pairs per group for this sublevel. At the top sublevel, we have one pair per group, and so [1,2], the result after pair stepping, becomes [2,3], which is the next pair. Similarly, we find [4,5] and [6,7].

In the second sublayer, we have two pairs per group, so we start out with [0,2], which directly steps to the next pair in that group as [1,3]. At the end of the first group we have [2,4] which is transformed to [4,6] as the first pair of the second group. Then we find [5,7] as the second pair of the second group.

In the third sublayer, we have four pairs per group, and just one group. So we start out with [0,4] and step to [1,5], then [2,6] and [3,7], at which time we have done the pairs, groups, and sublayers and so are done.

Pascal Mixing Routine

Here we show a routine in Pascal for mixing blocks of arbitrary power-of-2 size as selected at ciphering time. Dynamic size selection supports, for example, the general use of large (say, 64 byte) blocks with an end-of-message step down using at most one of each lesser block size until we reach "legacy" 8-byte (DES-size) blocks.

Note that the simple Balanced Block Mixing computations are included. Here we use the degree-8 polynomial 01a9h.

```
PROCEDURE ByFFTmix( VAR DatByAR; byct: WORD );
   { FFT-style byte mixing of 2**n contiguous bytes }
   { undo ByFFTmix with ByFFTmix; a self-inverse }
   VAR
      dat: ByteArray ABSOLUTE DatByAR;
      pairs, groups, el1, el2, group, pair, t: WORD;
   BEGIN

   groups := byct DIV 2;
   pairs := 1;

   WHILE (groups > 0) DO
      BEGIN

      el1 := 0;
      el2 := el1 + pairs;
      FOR group := 0 TO PRED(groups) DO
         BEGIN
         FOR pair := 0 TO PRED(pairs) DO
            BEGIN

            t := dat[el1] XOR dat[el2];  {  a +  b }
            t := t + t;                  { 2a + 2b }
            IF ((t AND $100) <> 0) THEN
               t := t XOR $1a9;          { 2a + 2b - p }
            dat[el1] := dat[el1] XOR t;  { 3a + 2b }
            dat[el2] := dat[el2] XOR t;  { 2a + 3b }

            Inc(el1);  Inc(el2);
            END;
         Inc(el1,pairs);
         Inc(el2,pairs)
         END;

      groups := groups DIV 2;
      pairs := pairs + pairs;
      END;

   END; {ByFFTmix}
```

C Mixing Routine

Here we show the same routine in C code for mixing blocks of arbitrary power-of-2 size as selected at ciphering time.

Note that the simple Balanced Block Mixing computations are included. Here we use the degree-8 polynomial 01a9h. The BYTE and WORD defines are as one might expect.

```
/*
 * FFT-style byte mixing of 2**n contiguous bytes
 *
 */
void
ByFFTmix( BYTE dat[], WORD byct )
{
   WORD  pairs, groups, group, pair;
   BYTE  *d1, *d2, t;

   groups = byct >> 1;
   pairs = 1;

   while (groups)
      {

      d1 = dat;
      d2 = d1 + pairs;

      for (group = 0; group < groups; group++)
         {

         for (pair = 0; pair < pairs; pair++)
            {

            t = *d1 ^ *d2;

            if (t & 0x80)
               t = (t + t) ^ 0xa9;
            else
               t = t + t;

            *d1++ ^= t;
            *d2++ ^= t;
            }

         d1 += pairs;
         d2 += pairs;
         }

      groups >>= 1;
      pairs <<= 1;
      }
}
```

Summary

We have developed a description and presented working routines in Pascal and C for a particular efficient FFT-style mixing pattern. This mixing pattern is used to extend byte-wide mixing across arbitrary power-of-2 size blocks.

It is also possible to consider patterns which may be re-used without change in each sublevel. But the major issue in a Mixing design is likely to be table storage, rather than mixing. Consequently, having only a single mixing sublayer may not have much of an effect on overall cost, but *will* constrain throughput.

There may be slightly more efficient ways to code this standard mixing pattern. But as it stands it does drop into 80x86 assembly language fairly well. Unfortunately, byte-level processing is not a big advantage of 32-bit code.

***Terry Ritter**, his **current address**, and his **top page**.*

***Last updated:* 1997-06-09**

Hardware Blowfish and Mixing Ciphers Compared

Ciphering on Chip: Performance vs. Cost

Terry Ritter

The goal of this exercise is to get some idea of the chip area required to encipher the same block under two different block cipher technologies, and to see what each implies in terms of throughput.

Here we sketch out preliminary chip gate counts and resulting performance for several hardware configurations.

How Much Chip Do We Need?

To get a general idea about how much of a chip these designs would consume, we can look at a few chip examples in current production:

- Static-memory devices typically lag dynamic memory sizing by somewhat more than a generation. Current Motorola SRAM chips include a range of 4mb (4 megabit) designs.
- Individual customizable gates are larger than static RAM cells. As data points, Altera has a 250k gate FPGA, and the average IBM ASIC is said to have been about 260k gates in 1996.

One might thus conclude that "a gate" is as large as 16 static RAM cells, on average, which seems a little high. But the actual value does not matter much here, because these designs are heavily dominated by RAM anyway.

A Mixing Cipher

The mixing ciphers described here have a fencing layer (a row of substitution tables), Balanced Block Mixing, fencing, mixing and fencing. These are conceptually very simple ciphers.

In a mixing cipher, the tables constitute primary keying and are set up by an external subsystem prior to ciphering.

A Balanced Block Mixer

The Mixing cipher uses Balanced Block Mixer or BBM components which realize:

```
a = 3x + 2y (mod 2, mod p)
b = 2x + 3y (mod 2, mod p).
```

This can be implemented as:

1. q := XOR(x, y);
2. a delayless shift of one bit left;
3. a conditional r := q or r := XOR(q, p); then
4. a := XOR(r, x) and b := XOR(r, y).

This is 32 XOR gates per byte-wide mixing, with an assumed 3 level gate delay.

These byte-wide mixings are applied across the 8 bytes to be mixed, 4 pairs at a time, in FFT-like patterns. Note that each of the BBM operations is separate and independent hardware, functioning in parallel with other hardware. After 3 mixing sub-layers, every one of the 8 input bytes is equally represented in each of the 8 output bytes.

64-bit Blowfish Realizations

Consider the 8 byte block cipher Blowfish: Here we have four 8-in 32-out tables, for a total of 4K (BYTES) or 32kb (bits) of RAM. The combining is 32 bits wide, and seems to have 4 operations, for 128 gates, plus 32 bits more in keying, for a total of 160 gates. This is for one round.

Note that the 32k RAM elements in one round dominates the 160-gate logic requirements by a factor of 200+, so we can simply ignore gate area with little error. And the 32k bits of static RAM in one round represents only 1/128th of a current mass-production 4Mb static RAM chip.

There are 16 rounds, and each uses exactly the same tables.

524kb Blowfish

For extreme speed, we can give each round its own tables, for a total of 64KB or 524kb (bits). This is still only 1/8th (about 12 percent) of a 4mb static RAM chip.

Since we can partition at will and pipeline to make our clock rate, the 524kb Blowfish realization has a throughput of 8 bytes per clock, with a latency of 16 look-ups and 16 4-gate-level combinings.

32kb Blowfish

A smaller Blowfish might have a single set of tables, representing 32kb of RAM, re-used for each of 16 rounds. 32kb is 1/128th (under 1 percent) of a 4mb static RAM chip.

The 32kb Blowfish has a throughput of 8 bytes every 16 clocks, with a latency of 16 look-ups and 16 4-gate-level combinings.

64-bit Mixing Realizations

A fully-expressed 8-byte-wide Mixing cipher needs 24 8-in 8-out tables, for a total of 6K (BYTES) or 48kb (bits). We also need 4 BBM's in each mixing sublayer, times 3 sublayers, times 2 overall mixings, for a total of 24 BBM's. At 32 gates each this is 3/4k gates.

Again note that the 48k RAM elements in the cipher dominate the 768-gate logic requirements by a factor of 64, so we can generally ignore gate area. In practice, the BBM design and layout probably would be optimized and not simply constructed from gates, so we can expect this to be tighter than average anyway.

48kb Mixing

For best speed, we simply realize each separate table, for 48kb of RAM. 48kb is about 1/85th (under 2 percent) of a 4mb static RAM chip.

The 48kb Mixing cipher has a throughput of 8 bytes per clock with a latency of 3 look-ups and 6 3-gate-level BBM mixings.

16kb Mixing

The next reasonable mixing system uses a single fencing row and one full mixing layer, and re-uses those. So we need 16kb for the tables, and only one mixing hardware. 16kb is about 1/256th (less that 1/2 of 1 percent) of a 4mb static RAM chip.

The 16kb Mixing cipher has a throughput of 8 bytes per 3 clocks, with a latency of 3 look-ups and 6 3-gate-level BBM mixings.

Comparisons

Compared to 524kb Blowfish, the 48kb Mixing cipher is about 1/10th the size, has an equivalent throughput, and perhaps 1/2 the latency.

Compared to 32kb Blowfish, the 16kb Mixing cipher is about 1/2 the size, has over 5 times the throughput, and perhaps 1/4 the latency.

Terry Ritter, his current address, and his top page.

Last updated: 1997-03-27

sci.crypt #46705 (9 more)
Path: news.io.com!io.com!not-for-mail
From: ritter@io.com (ritter)
Newsgroups: sci.crypt

Subject: Fencing and Mixing Ciphers
Date: 16 Jan 1996 19:31:11 -0600
Organization: Illuminati Online
Lines: 242
Message-ID: <4dhjgv$rn5@pentagon.io.com>
NNTP-Posting-Host: pentagon.io.com

From my previous work on fencing and mixing ciphers I have realized a family of five interesting block cipher architectures in 64-bit, 128-bit, and 256-bit widths.

You won't find this sort of cipher in the crypto texts.

These working prototypes are intended to explore a range of "Fencing" and "Balanced Block Mixing" designs, and provide some crude performance comparisons. This is patent-pending technology.

My Technical Terms

* By "fencing" I mean an array of typically byte-width substitutions. Each 256-byte table corresponds to a keyspace of 1684 bits (provided, of course, that the rest of the cipher will hide the arrangement of the table).

* By "balanced block mixing" I mean a structure which generally takes two input sub-blocks to two output sub-blocks and provably propagates a change in *either* input sub-block to *both* output sub-blocks. (This is sufficient to provide provable avalanche in a block cipher.)

I show the structure for the "4x" or "256-bit block" versions. The large block is used for most ciphering; one each 2x and 1x block may be used at the end of the file to reduce ciphertext expansion to that of a 64-bit block.

Performance measurements occurred for RAM-drive ciphering of a 750K file on a P90 under DOS 6.22, with single-pass shuffles.

Fenced DES (init = 31ms, ciphering = 181 KB/sec)

```
                              INPUT
<------------------------- 256 bits -------------------------->

S S S S S S S S S S S S S S S S S S S S S S S S S S S S S S S S  fencing
--------------------------------x-------------------------------  mixing
---------------x--------------- ---------------x---------------  mixing
------DES------ ------DES------ ------DES------ ------DES------  DES
---------------x--------------- ---------------x---------------  mixing
--------------------------------x-------------------------------  mixing
S S S S S S S S S S S S S S S S S S S S S S S S S S S S S S S S  fencing

<------------------------- 256 bits -------------------------->
                               OUTPUT
```

Here we have 32 input and 32 output byte-substitutions (S) across a 256-bit block. We also mix (---x---) 128-bit and 64-bit sub-blocks using linear Balanced Block Mixing. Because all data flows through DES, the cipher cannot be weaker than DES. Each substitution layer is protected by DES and so apparently cannot be exposed without breaking DES and the other substitution layer.

Each 256-byte substitution table is keyed by shuffling under a cryptographic RNG initialized from a User Key. That RNG also produces 4 separate random DES keys. Normally, the keyspace of this sort of cipher is limited by the size of the RNG used to key the substitutions, and it is easy to have a true keyspace of thousands of bits.

The ability to attack the keying RNG depends upon developing the state in one or more of the substitutions, then inferring the RNG sequence. But inferring the RNG sequence can be made difficult or impossible by double-shuffling each substitution. It is not at all clear how an attacker could develop the correct state of any substitution in the first place. Even a single bit error in any input table is guaranteed to produce avalanche, so the extent of solution of these tables cannot be distinguished.

(The Fenced DES cipher was described on sci.crypt almost two years ago, in a post I have archived as

http://www.io.com/~ritter/NEWS/94042901.HTM

).

Fenced Tree (init = 45ms, ciphering = 113 KB/sec)

```
S S S S S S S S S S S S S S S S S S S S S S S S S S S S S S S S  fencing
--------------------------------x-------------------------------  mixing
---------------x--------------- ---------------x---------------  mixing
---------------x--------------- ---------------x---------------  mixing
---x---  ---x---  ---x---  ---x---  ---x---  ---x---  ---x---  ---x---  mixing
-x- -x- -x- -x- -x- -x- -x- -x- -x- -x- -x- -x- -x- -x- -x- -x- -x- -x-  mixing
S S S S S S S S S S S S S S S S S S S S S S S S S S S S S S S S  fencing
-x- -x- -x- -x- -x- -x- -x- -x- -x- -x- -x- -x- -x- -x- -x- -x- -x- -x-  mixing
---x---  ---x---  ---x---  ---x---  ---x---  ---x---  ---x---  ---x---  mixing
-------x--------  -------x--------  -------x--------  -------x--------  mixing
---------------x--------------- ---------------x---------------  mixing
--------------------------------x-------------------------------  mixing
S S S S S S S S S S S S S S S S S S S S S S S S S S S S S S S S  fencing
```

This is similar to Fenced DES, but with three more Balanced Block Mixing layers (for a total of five) and another fencing layer instead of the DES layer.

Fenced Quad (init = 45ms, ciphering = 398 KB/sec)

```
S S S S S S S S S S S S S S S S S S S S S S S S S S S S S S S S  fencing
============================================================  lin-FFT
======= ======= ======= ======= ======= ======= ======= =======  lin-FFT
S S S S S S S S S S S S S S S S S S S S S S S S S S S S S S S S  fencing
======= ======= ======= ======= ======= ======= ======= =======  lin-FFT
============================================================  lin-FFT
S S S S S S S S S S S S S S S S S S S S S S S S S S S S S S S S  fencing
```

Here we have two full linear "FFT" operations between three fencing layers. Note that many different mixing arrangements will produce equally acceptable mixing.

Each "FFT" "butterfly" operation is a Balanced Block Mixer of appropriate width. Basically, this design assumes that mixing with 32-bit sub-blocks is likely to be faster than mixing with 8-bit sub-blocks. Consequently, the mixing is broken down into 32-bit "FFT" layers and 8-bit "FFT" layers.

Fenced FFT (init = 45ms, ciphering = 413 KB/sec)

```
S S S S S S S S S S S S S S S S S S S S S S S S S S S S S S S S  fencing
============================================================  lin-FFT
S S S S S S S S S S S S S S S S S S S S S S S S S S S S S S S S  fencing
============================================================  lin-FFT
S S S S S S S S S S S S S S S S S S S S S S S S S S S S S S S S  fencing
```

Here we also have linear "FFT-like" mixing between three fencing layers. But here, every "FFT" "butterfly" operation is a byte width Balanced Block Mixer.

Fenced OLS (init = 76ms, ciphering = 372 KB/sec)

```
                              INPUT
<------------------------- 256 bits -------------------------->

============================================================  nl-FFT

<------------------------- 256 bits -------------------------->
                               OUTPUT
```

Here we have non-linear FFT-style mixing, and no fencing layers at all. Each "FFT" "butterfly" is a byte-width Balanced Block Mixer composed of a keyed pair of orthogonal Latin squares of order 256.

This prototype uses a single 128K randomized table, with a keyspace of 3368 bits. But only memory space and initialization time prevents us from using a different table at every mixing node, thus producing an astronomical keyspace.

Avalanche

It is interesting to contrast the avalanche characteristics in this class of mixing cipher with that of the ideal block cipher, which is a huge keyed Simple Substitution. In the ideal case, a one-bit change in the input block has the potential to change just a single bit in the output block. But in these mixing ciphers, a one-bit change in the input block will change at least one bit *in every mixing output*.

However, this particular case is so rare that we could probably never find such a value in the ideal cipher (even though such values must exist!). Nor could we expect to see the difference in practical statistical tests. So here we seem to have designs whose known deviations from the ideal occur in areas which we cannot measure.

Complexity

These ciphers are simultaneously more simple and also more complex than conventional ciphers: These ciphers have far simpler structures than, say, "round" based Feistel ciphers, and rely on simple, *provable* mixing characteristics. (See, for example,

http://www.io.com/~ritter/FENCED.HTM (24K)

). But these ciphers have far more "state" than conventional (mainly "algorithmic") ciphers. These ciphers also *key* the internal state (as opposed to using the same state for all users).

I expect that the large-state approach is generally more secure than the algorithmic approach in that a large keyed state *demands* a large amount of investigation to define that state. In contrast, a mainly algorithmic approach seems perhaps more vulnerable to unknown algorithmic attack.

Having a clean, simple structure should mean that any attacks based on that structure should also be fairly clean and apparent. It *should* be easier to have confidence in these ciphers than in complex architectures which have plenty of opportunity to hide attack techniques.

There is an additional storage cost for the large-state approach, but, for these sizes, it seems almost irrelevant. This state does require a keyed initialization which takes longer than the usual algorithmic cipher. Normally, keyed initialization is arranged to occur infrequently.

Dynamic Keying

Zero-overhead block-by-block dynamic keying is available by XORing the input block with a like-size key. I would not expect this to increase native strength, but I would expect it to defeat attacks which rely upon knowing the input to the cipher over many blocks. As long as the cipher itself remains unbroken, this sort of dynamic key would seem to be well hidden.

Some moderate-overhead dynamic keying is also available in Fenced DES by changing one or more of the DES keys.

At this time I would resist adding internal XOR keying layers, with the thought that such layers might expose otherwise hidden internal operations.

Your Comments Wanted

All comments on these designs, positive or negative, will be appreciated. Serious financial support is also needed. Companies interested in custom ciphers based on these or other technologies should contact me directly.

References

http://www.io.com/~ritter/NEWS/LBDNEWS.HTM#LBDBBM
http://www.io.com/~ritter/BBM.HTM (13K)
http://www.io.com/~ritter/NEWS/LBDNEWS.HTM#LBDFD
http://www.io.com/~ritter/FENCED.HTM (24K)

Terry Ritter Ciphers By Ritter http://www.io.com/~ritter/

The Fenced DES Cipher

Stronger Than DES But Made From DES

Terry Ritter

In the recent past, the U.S. Data Encryption Standard (DES) has been the single widely-accepted cryptographic standard. However, continued advances in computation speed have now placed DES in danger of attack. For the future we need a stronger cipher, hopefully one which takes much less computation than Triple-DES. One approach is to perform four cipherings of DES over a large block with, and mix these ciphertexts, high and low the mixing, so that each ciphering contributes to each bit of the block. This approach uses the well-known strength of DES to produce a much stronger "fenced" version of DES, which I call Fenced DES (by analogy as fast as DES itself. Using the fix version with smaller 2x and 1x versions means that data will not expand any more than DES itself, thus generally providing a DES replacement based on DES.

Contents

- Introduction
  - DES Continues to Weaken, The Triple-DES Alternative, Other Alternatives, Replacing DES with DES
- Overview
- Large Block DES
  - 1x Fenced DES: Table Mixed DES, 2x Fenced DES: Balanced Block Mixing, 4x Fenced DES
- Strength of 1x Fenced DES
  - With Known Substitutions, With Known DES Key, Conclusion
- Strength of 4x Fenced DES
  - Minimum Strength, Conclusion
- Attacks
  - Exhaustive Search, Codebook, Meet-in-the-Middle, Differential Cryptanalysis
- Conclusions
- NEW! RECENT ATTACKS AND RESPONSES

Introduction

DES Continues to Weaken

The U.S. Data Encryption Standard (DES) was defined almost two decades ago. During this period, unprecedented advances in computational machinery have improved the ability to attack the cipher. Although DES was never designed to be applied to implement the cipher as well as it attack it, DES has a keyspace which is fixed and limited. Even though DES can be made to operate faster, it cannot be any stronger than it was designed to be almost 30 years ago. Relative to available attack technology, the cipher is far weaker than it was, and continues to weaken.

DES has become vulnerable to modern technical attacks. It now appears that a substantial capital investment in engineering development and equipment could construct a "DES breaking" machine. Such a machine could search the entire DES keyspace in a few hours.

Would all users need to abandon DES if it could be penetrated at will by governments, corporations and organized crime? One can certainly argue that most data do not need ultimate protection. But when a DES-protected message finally is recovered, it is the exception of that expense argue that the machine will be kept busy if at all possible. Thus, DES attacks could become relatively fast and cheap. Businesses which currently protect very expensive and marketable secrets with DES should take immediate notice.

The Triple-DES Alternative

To maintain earlier levels of security, DES must be replaced with a stronger cipher. The one obvious alternative to DES is a simple construct both from DES cipher *Triple-DES*. Triple-DES, while generally being thought of as "strong enough," also unfortunately requires three times the computation of normal DES.

Because every security system is required to provide more benefit than its cost, raising costs by a factor of three (compared to normal DES) is a significant issue. Such costs could dangerously delay the necessary extension of ordinary DES.

Other Alternatives

New ciphering algorithms are often challenged to "prove" they are stronger than DES. Since it is **impossible** to measure the "strength" of a cipher (and there is no absolute proof of strength for any cipher), such challenges are simply unfair and often outright ridiculous. On the other hand, DES itself is well-known and accepted (despite having no proof of strength), so there seems to be some advantage in using DES as a component in a stronger cipher.

Replacing DES with DES

We want to build a stronger cipher. Perhaps we can do this by using DES as a component. The goal would be to upgrade the strength of DES while using substantially less computation than Triple-DES. A sufficiently clean and "believable" design might even achieve a "derivative certification," and thus inside the huge expense and delay involved in certifying a totally-new cipher.

Overview

Modern block ciphers seek to emulate a substitution table for larger than anything which can be realized. Doing this properly is tricky (as I well know). But all ciphers are built from smaller components, and the simple complexity of these architectures means that, eventually, we understand how they work. We can now select some of these principles and build a wide block cipher based on DES.

It is unfortunately impossible to measure the strength of a cipher, but we can measure related quantities. Perhaps the more useful quantity in a block cipher is *overall diffusion* or "avalanche." For avalanche, change one single input bit and see whether about half the output bits have changed. Now, if we change the input block by even a single bit, each bit in the output block should change with probability 0.5. If we change different input bits, we will get different results, but still, about half of the output bits should change. If we perform this experiment for many different input values, we can measure the ability of the cipher mechanism to diffuse changes across the entire block.

The essence of building a wide block cipher from smaller components is *mixing* between the components. It turns out that we can develop a Balanced Block Mixing component which, when used with a substitution table or block cipher, provably assures such mixing. With this new component we can construct multiple DES cipherings together into a larger block. And, based on universally accepted assumptions about DES, we can prove that the resulting structure must have overall diffusion. While this is not "strength," it is related to strength, and having clear proof of a major ciphering property is very unusual. DES itself may have no such proofs.

Large Block DES

I have proposed a DES alternative which I call Fenced DES (because DES appears to be "fenced off"). We will later develop versions with larger block widths:

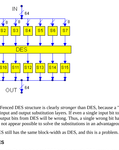
1x Fenced DES

Here is the 1x Fenced DES construct:

```
S S S S S S S S
 - - - - - - - -
S S S S S S S S
```

Each of the "S" characters represents an "eight-bit-wide" (256-byte) substitution table shuffled by a Random Number Generator (RNG) initialized from some key. The tables will be initialized before ciphering, and will be used for ciphering.

First note that all of the message information flows through DES itself. This means that the overall cipher cannot possibly be weaker than DES. This is a particularly important point for any new cipher based on DES.



Moreover, the 1x Fenced DES structure is clearly stronger than DES, because a "known plaintext" attack requires knowing the actual input and output values at the DES cipher itself, and these values are hidden by the input and output substitutions. If even a single input bit to DES is wrong (that is, if even one value in our substitution tables is off by even one bit), DES will "avalanche" and produce a completely different result. Thus, to attack the DES cipher we must work through the same statistical effects as a merely-wrong value; this means that it is not possible to know when the input value is "close," or to use any other shortcut because the substitutions are unknown.

But 1x Fenced DES still has the same block-width as DES, and this is a problem. A block size adopted somewhat adequate in the mid-70's is rapidly becoming a manipulable quantity.

Table Mixed DES

Suppose we want to handle a double-sized block at nearly DES speeds, how do we do it? Well, we certainly must mix all the input bits, so that a change in even one bit will affect both DES operations. In addition, each and every output bit should be an invertible function of both DES operations. Here is one approach:

```
 | | | | | | | | |   | | | | | | | |
 | | | | | | | | |   | | | | | | | |
 | | | | | | | | |   | | | | | | | |
```

United States Patent 5,727,062

Variable Size Block Ciphers

Terry Ritter

Ciphers for handling blocks of arbitrary size to the byte, as dynamically selected at ciphering time, while retaining good block cipher properties

For informational purposes only. Taken from the original Application and Amendment files which were sent to the PTO in printed form. At the PTO, the documents were manually transcribed into the printed patent. Here, the files were converted to HTML, partly by hand. For these reasons this version may differ, somewhat, from the printed patent. If there is any question, see an actual printed patent copy from the PTO, with the Certificate of Correction, if any.

Contents

- Title
- References
- Abstract
- Drawings
- Body
  - Cross-Reference to Related Applications
  - Technical Field
  - Background
  - Prior Art
  - Substitution-Permutation Ciphers
  - Feistel Ciphers
  - Modes of Operation
  - Others
  - No Literature on Variable Size Block Ciphers
  - Casual comments On Improving Stream ciphers
  - No Literature on Variable Size Layers
  - No Literature on Dedicated Diffusion Layers
  - No Known Literature on Effective Block Ciphers with Few Rounds
  - No literature on Use of New Combiners in Block Ciphers
  - Literature Reports Difficulty in Doubling Fixed Block Size
  - Objects and Advantages
  - Objects
  - Fundamental Speed Advantage
  - Fundamental Strength Advantage
  - Better Fit To Existing Systems
  - Better Fit to Inherently Variable-Size Devices
  - Speed Advantage From Large Blocks
  - Strength Advantage From Large Blocks
  - Unexpectedly Supports A New Form of Keying
  - Unexpectedly Supports a Secure Authentication Code
  - Supports Other Forms of Dynamic Keying
  - Inherently Supports Testable Small Versions
  - Inherently Supports Easier Analysis
  - Need Not Expand Data At All
  - Summary of the Invention
  - Layers
  - Arbitrary Size
  - Keying
  - Overall Diffusion
  - Brief Description of the Drawings
  - Description of The Presently Preferred Exemplary Embodiments
  - Fig. 1--Simple Exclusive-OR Diffusion
  - Fig. 2--Diffusion Across Layers
  - Fig. 3--Balanced Block Mixing for Diffusion
  - Fig. 4--Latin Square Combined Confusion and Diffusion
  - Operation
  - Fig. 1--Simple Exclusive-OR Diffusion
  - Fig. 2--Diffusion across Layers
  - Fig. 3--Balanced Block Mixing for Diffusion
  - Fig. 4--Latin Square Combined Confusion and Diffusion
  - Glossary
  - Conclusion, Ramifications and Scope of Invention
- Claims

United States Patent	**5,727,062**
Ritter	**Mar. 10, 1998**

Variable size block ciphers

Inventors: **Ritter; Terry F.** (2609 Choctaw Trail, Austin, TX 78745).

Appl. No.: **499,024**
Filed: **Jul. 6, 1995**

Intl. Cl. :	**H04L 9/18**, H04L 9/00
Current U.S. Cl.:	**380/37**
Field of Search:	380/37, 42

References Cited

U.S. Patent Documents

3,798,359	Mar., 1974	Feistel	178/22
3,798,360	Mar., 1974	Feistel	178/22
3,962,539	Jun., 1976	Ehrsam et al.	178/22
4,078,152	Mar., 1978	Tuckerman	178/22
4,157,454	Jun., 1979	Becker	178/22
4,275,265	Jun., 1981	Davida et al.	178/22.09
4,316,055	Feb., 1982	Feistel	380/37
4,751,733	Jun., 1988	Delayaye et al.	380/42
4,979,832	Dec., 1990	Ritter	380/28
5,003,597	Mar., 1991	Merkle	380/37
5,214,703	May, 1993	Massey et al.	380/37
5,231,662	Jul., 1993	van Rumpt et al.	380/9

Other References

Gutmann 1974 Excerpt from 400KB Secum File System Documentation.

Moyes, Rubin, Murray, Johnson, Scott, Schneier Messages from Sci. Crypt (Internet Messages from Jan. 1995).

Primary Examiner: Tarcza; Thomas H.
Assistant Examiner: White; Carmen D.

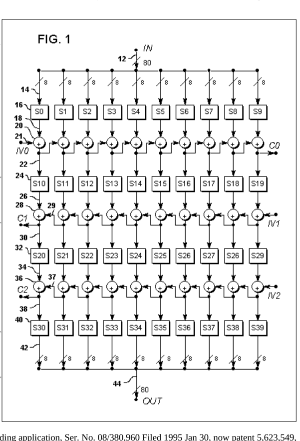
ABSTRACT

Data blocks of various size can be processed in fast, effective block cipher mechanisms. Variable size confusion layers and variable size diffusion layers combine to form variable size block ciphers. Block size can set (typically byte-by-byte) at design time or dynamically during operation.

The embodiment of Fig. 1 consists of horizontal layers alternating between confusion and diffusion. Here, each confusion layer, such as the layer including byte-wide substitution operation (16), uses keyed substitution tables which are initialized prior to operation. Simple and fast diffusion layers, such as the layer including byte-wide exclusive-OR (20), generally diffuse in just one direction. Consequently, multiple diffusion layers are usually required, such as those including byte-wide exclusive-OR operations (28) and (36). This kind of diffusion is unusual in a block cipher because it is extremely weak. However, because it can cover the entire block in a single layer, it is unexpectedly advantageous when combined with confusion layers.

The resulting variable size block ciphers are unusually fast and can be made arbitrarily strong. These ciphers can better fit existing applications, reduce overhead, and directly support large and efficient data blocks. Block size can be expanded dynamically, with good overall diffusion, without adding processing layers. These ciphers support better testing and analysis than previous designs, and also support unexpected new techniques for dynamic keying and authentication.

7 Claims, 4 Drawing Figures



DRAWINGS

FIGURE 1: Data-flow in an 80-bit block cipher using keyed byte-wide substitutions for confusion and byte-wide exclusive-OR operations for diffusion.

FIGURE 2: A cipher similar to Fig. 1, with improved diffusion and strength.

FIGURE 3: An 80-bit block cipher which uses byte-wide Balanced Block Mixer operations for diffusion.

FIGURE 4: An 80-bit block cipher which uses Latin square combining operations for simultaneous confusion and diffusion.

BODY

VARIABLE SIZE BLOCK CIPHERS

Cross-Reference to Related Applications

Some variable size block ciphers use the Latin Square Combiner and Extractor or the Balanced Block Mixer of my co-pending application, Ser. No. 08/380,960 Filed 1995 Jan 30, now patent 5,623,549, granted 1997 Apr 22.

Technical Field

This invention relates to the art of cryptography, the secure transmission or storage of information; more particularly, this invention relates to constructing or enhancing mechanisms for ciphering data in blocks of variable size.

Background

A cipher is a way to hide the information in a message when that message may be intercepted. A typical cipher can translate a message into a multitude of different forms, as selected by a key, and then translate any of those forms back into the original, given the same key. Typically, each user has a particular key which specifies an arbitrary but particular intermediate form for their messages. Subsequently, these messages can be exposed by anyone with the same key and cipher, but no one else.

An effective cipher reduces the problem of protecting information to that of enciphering the information and giving keys only to authorized individuals. Controlling key distribution for authorized access is a difficult but well-known issue which is not improved in this application.

Any cipher may be attacked by an opponent in an attempt to gain the hidden information without the use of the correct key. Even though such attacks can be extremely difficult, time-consuming, and costly, many apparently secure designs have been attacked and their protection defeated. Unfortunately, there is no known way to measure the strength of a particular design.

The best-known block cipher is the U.S. Data Encryption Standard or DES, which was adopted as a Federal standard on November 23, 1976. DES is commonly used to protect banking data and other communications, and is often the reference by which other cipher mechanisms are judged.

Since 1976, unprecedented advances in the use of exposed communications (e.g., satellite communications, the Internet, etc.) have increased the need for a secure cipher. But equally unprecedented advances in computational technology now threaten to overwhelm the strength of DES. These advances now make it possible to build a machine which can search all possible DES keys in a reasonable time. Because DES could now be successfully attacked, a new cipher design is needed.

There are many designs for ciphering mechanisms. As one might expect, some may be more secure than others, some are faster than others, while others are perhaps more understandable. Since few if any designs are simultaneously superior in every area, there is a continuing need for ciphers with particular characteristics.

The new ciphering characteristic of variable block size (typically variable in byte-size steps) of this application has many advantages. A variable size block cipher can: be a better match to existing systems; be a better match to inherently variable-size data sources; have improved speed; have the potential for zero data expansion; provide the first practical support for new forms of dynamic keying and authentication; have clarity of construction and simpler analysis; and include the ability to downsize the live cipher for testing.

Prior Art

Perhaps the best single source of information on the huge variety of block cipher designs is the book Applied Cryptography by Bruce Schneier, published by John Wiley & Sons, Inc. in 1994. But here we discuss the patent record:

Substitution-Permutation Ciphers

U.S. patent 3,798,359 titled Block Cipher Cryptographic System issued 1974 Mar 10 to Horst Feistel (IBM) gives a detailed design which has a substitution-permutation structure. (Claim 1 has: "means for carrying out a plurality of substitution operations" and "means for rearranging the combination of binary representations in said block.") Here, "rearrangement" means "permutation." In contrast, the present application uses variable size diffusion layers instead of the permutation layer of the '359 patent. Because of the fixed nature of the permutation (FIG. 5B of that patent), the '359 patent cannot accommodate the variable size blocks of the present application.

U.S. patent 4,275,265 titled Complete Substitution Permutation Enciphering and Deciphering Circuit issued 1981 Jun 23 to George Davida and John Kam teaches another form of substitution-permutation ciphering. This design (FIG. 1 of that patent) uses three substitution layers, and the interconnections between these layers are the permutation connections. The "substitution boxes" in the '265 patent are not general substitution tables containing keyed arbitrary permutations of the possible values, but instead are just two fixed mappings (FIG. 2 of that patent). In each box, three data bits and one key bit are input, but and only three result bits are produced; the key bit selects between two particular ("complete") pre-defined three-bit-wide mappings. In contrast, the present application uses full substitution tables which are shuffled or keyed in arbitrary (instead of particular) arrangements. Moreover, the particular permutation structure in the '265 patent (the connections between layers of FIG. 1 of that patent) cannot handle blocks of different size without significant changes. Thus, the approach of the '265 patent cannot handle the variable size blocks of the present application.

U.S. patent 4,751,733 titled Substitution Permutation Enciphering Device issued 1988 Jun 14 to Bernard Delayaye et. al. (Thomson-CSF) describes a classic substitution-permutation cipher. Although it does use byte-width substitutions, the rigid structure of the bit-permutation connections (FIG. 5 of that patent) make it unsuitable for processing the variable size blocks of the present application.

U.S. patent 5,231,662 titled Method and Device for Enciphering Data to be Transferred and for Deciphering the Enciphered Data, and a Computer System Comprising Such a Device issued 1993 Jul 27 to Herman van Rumpt and Benny Kwan (Tulip Computers International) teaches yet another form of substitution-permutation cipher. The Summary (col. 2 of that patent) teaches that one goal of this approach is to provide a cipher whose block width is equal to the width of the data bus in a general purpose computing machine. While that patent claims encipherment for data words of "n" bits, claim 1 (col. 6, line 60) clearly states that n is a "pre-defined integer." Thus, the approach of the '662 patent cannot handle the variable size blocks of the present application.

U.S. patent 4,157,454 titled Method and System for Machine Enciphering and Deciphering issued 1979 Jun 5 to Wolfram Becker (IBM) teaches the ciphering of variable length data fields (Summary, col. 2, line 44 of that patent) in a substitution-permutation cipher. Nevertheless, the approach of the '474 patent cannot be said to handle the variable size blocks as defined in the present application:

Simple forms of permutation, such as the rotation suggested in the '474 patent (col. 4, lines 20-21), distribute bits slowly. For example, if we rotate a data block (by any number of bits which is not a multiple of the element size), each previous element can place bits in (and thus diffuse into) two adjacent subsequent elements. But by further rotating that block in the next round, the best we can do with now two adjacent elements is to place bits in three subsequent adjacent elements, not four. This implies a need for n execution rounds for a block of n elements for even the most basic diffusion. This need to vary the number of rounds to achieve the overall diffusion required in a block cipher does not qualify the approach in the '474 patent as a variable size block cipher as defined in the present application.

In contrast, in tests with 200-byte blocks, the examples of the present application provide overall diffusion in just 3 layers. For a similar 200-byte block, the design of the '474 patent would seem to require 199 block rotations and substitutions. This is a dramatic practical difference.

It is worthwhile to point out that substitution-permutation ciphers inherently need multiple rounds to achieve overall diffusion, and inherently the number of rounds must vary with block size: Clearly, each substitution operation does diffuse bits, but only within the substituted element. And each permutation operation does diffuse bits, but only between distinct elements (this is a bit re-distribution). So the result of any one substitution can at most affect as many elements as there are bits in an element. This means that the permutation approach simply cannot diffuse larger blocks without additional rounds. In marked contrast, the diffusion layers of the present invention allow each substitution result to affect arbitrarily many subsequent substitutions (in one linear direction) in a single diffusion layer.

If we seek to increase block size in conventional substitution-permutation ciphers, we must find permutations which evenly distribute bits for each size block we support. Unless we also increase the element size, more rounds are required to maintain the overall diffusion demanded in a block cipher. And if we increase the element size, we have to handle substantial increases in the size of all substitution tables. These facts constrain most substitution-permutation ciphers to a fixed design size and number of rounds.

In contrast, the variable size diffusion layers of the present invention can diffuse information across an entire block (typically, one direction at a time) in just one layer. Just two such layers typically will diffuse any bit to any other bit. Just three such layers can produce good overall diffusion which is essentially independent of block width. Other diffusion layers may be used to further improve diffusion and add strength.

Feistel Ciphers

U.S. patent 3,962,539 titled Product Block Cipher System for Data Security issued 1976 Jun 8 to William Ehsam et. al. (IBM) appears (FIG. 8 of that patent) to use what is now known as the conventional Feistel architecture like that used in DES. This approach uses fixed substitution tables, instead of the keyed or shuffled tables of the present application. Moreover, the fundamental Feistel architecture of the '539 patent works only with some fixed ratio between internal element size and block size. (In both '539 FIG. 8 and DES the internal elements are half-blocks.) Thus the approach of the '539 patent cannot handle the variable size blocks of the present application.

U.S. patent 5,003,597 titled Method and Apparatus for Data Encryption issued 1991 Mar 26 to Ralph Merkle (Xerox) describes a round-oriented Feistel cipher variant. The patent does teach keyed or shuffled substitutions as used in the present application. The patent also teaches keying by combining an "Aux Key" with the data across the width of the block, which is also a dynamic keying option in the present application. But the '597 patent is claimed for fixed 64-bit data blocks, thus making it unsuitable for processing the variable size blocks of the present application.

All Feistel ciphers can at best handle blocks which are some small integer multiple of a power-of-two. Typically, Feistel ciphers such as DES use two sub-blocks of half size each. This architecture is fundamentally unsuitable for processing the variable size blocks of the present application.

Modes of Operation

U.S. patent 3,798,360 titled Step Code Ciphering System issued 1974 Mar 19 to Horst Feistel (IBM) appears to cover what we would now call a "mode of operation" in that it uses an internal block cipher (22 in FIG. 1 of that patent). The '360 patent does state (col. 3, lines 20-22) "the principles of this invention are not limited to any particular data feed register size." But it also says (col. 3, lines 11-14) "At some point in time, when sufficient data is accumulated in the data block of proper dimension for enciphering," clearly implying that there is some "proper dimension." Thus, the '360 patent apparently does not handle the variable size block ciphers of the present application.

U.S. patent 4,078,152 titled Block-Cipher Cryptographic System with Chaining issued 1978 Mar 7 to Bryant Tuckerman (IBM) again does not cover a cipher per se, but instead covers the well-known chaining mode of operation for use with any existing block cipher (such as DES). One purpose of this (col. 3, lines 29-33 of that patent) is to avoid the natural consequence of a block cipher, where identical input blocks will produce identical ciphertext. This produces a one-way diffusion in the same sense as a stream cipher, but not as a layer in a block cipher, not in combination with a confusion layer, and makes no pretense of providing or intending to provide overall diffusion. A chaining mode is not a block cipher in the sense of the present application, and the '152 patent does not handle the variable size block ciphers of the present application.

Others

U.S. patent 5,214,703 titled Device for the Conversion of a Digital Block and Use of Same issued 1993 May 25 to James Massey and Xuejia Lai (Ascom Tech AG) teaches the well-known IDEA cipher. This is a structure inherently limited to blocks of 64 bits or less ("m = 4, 8, or 16" in col. 4 line 69 of that patent) by the combining operation of multiplication mod 2m+1. Since 216+1 (and 28+1`and 24+1) is prime, it can creates a mathematical field, but 232+1 is not prime, so the same approach does not scale up. Thus, the '703 patent does not handle the variable size block ciphers of the present application

No Literature on Variable Size Block Ciphers

As far as is known, the literature contains no examples or discussion of the variable size block ciphers of the present application. There is also no known discussion of the advantages of such a cipher.

There are many stream ciphers, which do handle variable amounts of data, but do not provide overall diffusion. In a block cipher, if we change any single bit of the input data, we expect to see about half of the output bits change.

There are also many block ciphers, which do provide overall diffusion, but conventional block ciphers always have some particular fixed block size. It is very difficult to change the block size in conventional designs to end up with a good larger cipher, whether the motive be convenience, performance, or strength.

Because of the obvious advantages inherent in a variable size block cipher, the lack of such implementations in the literature indicates that variable size block cipher design is clearly unobvious.

Casual comments On Improving Stream ciphers

There have been Internet comments in the Usenet News sci.crypt group which suggest using a stream cipher first forward, then backward, to improve its diffusion. Other comments suggest a carry of accumulated information from the first pass to the second pass.

My experiments with the fast and weak variable size layers described in this application indicate that not one but two subsequent passes are generally needed to approach the overall diffusion requirements of a block cipher. This finding may or may not apply to stream ciphers in general.

No Literature on Variable Size Layers

All ciphers are made from smaller components, and many block ciphers have operations in layers of some form or other, but it is unusual to find a block cipher design in which each layer can be easily extended (in element-size steps) to arbitrary size.

The variable size block ciphers of this application typically use two types of layer, which I call confusion and diffusion. A confusion layer might consist of an array of substitution elements, each of which might be unique (for example, shuffled under the control of a key). A substitution layer of some sort is an integral part of well-known substitution-permutation architectures, but these are not variable size block ciphers.

A diffusion layer would be some sort of mixing layer which would propagate changes across the block. Such a layer would be useless within the context of a substitution-permutation design, making this clearly contrary to the prior art. Fast and simple diffusion layers do tend to propagate in only one direction, so multiple such layers are usually needed.

Some layers perform both confusion and diffusion. But because there is no literature on variable size block ciphers, there is also no literature on the variable size layers needed to build such ciphers.

No Literature on Dedicated Diffusion Layers

The variable size diffusion layers of this application are particularly intended to provide diffusion without necessarily providing cryptographic strength. These are layers which are normally considered too weak to use in the context of cryptography, and so would be unexpected in conventional block cipher design. Again, this is contrary to prior art.

In contrast, conventional Feistel designs like DES apparently depend upon the uncertain diffusion effects of a random function. The variable size block cipher technology of this application provides improved diffusion, in fewer layers, and an overall more efficient cipher than previous technologies.

No Known Literature on Effective Block Ciphers with Few Rounds

The variable size block technology of this application produces good overall diffusion in fewer rounds than conventional Feistel block cipher designs like DES. The variable size diffusion layers of this application are guaranteed to affect even the most distant byte of the block (in one direction at a time). In contrast, Feistel block cipher designs apparently rely on the uncertain diffusion effects of a random function which affects at most a full sub-block at once. Substitution-permutation designs are limited to re-distributing the limited number of bits in each element to each other element, and so cannot diffuse a large block quickly.

If we consider a variable size block cipher "round" to be a confusion layer and a diffusion layer, the technology of this application produces good overall diffusion in 4.5 rounds. In contrast, the technology used in DES apparently requires 8 rounds for similar diffusion (Schneier p. 237). Both technologies may require some additional processing for strength, but only the variable size block ciphers of this patent can handle large blocks without additional layers or rounds.

No literature on Use of New Combiners in Block Ciphers

It is unique to find a block cipher which uses Latin squares, Balanced Block Mixers, or Dynamic Substitution.

Dynamic Substitution is the topic of patent 4,979,832 issued 25 Dec 1990 titled Dynamic Substitution Combiner and Extractor.

Balanced Block Mixers are a topic in co-pending application number 08/380,960 filed 1995 Jan 30 titled Cipher Mechanisms with Mixing and Balanced Block Mixers.

Latin Square Mixing is another topic in co-pending application number 08/380,960 filed 1995 Jan 30 titled Cipher Mechanisms with Mixing and Balanced Block Mixers.

Literature Reports Difficulty in Doubling Fixed Block Size

Schneier reports (pp. 255-256) a proposal to double the block size of the IDEA cipher from 64 bits to 128 bits. However, the IDEA cipher depends upon a combining operation of a specific size (16 bits, using the field based on the prime $2^{16}+1$, and $2^{32}+1$ is not prime). As is the usual case with a conventional block cipher, even doubling the size of the block is difficult, let alone supporting byte-by-byte expansion.

Objects and Advantages

Objects

One objective of this invention is to provide constructions for practical, efficient, and effective block ciphers which handle data blocks of various size. It is a further objective to provide block cipher constructions which are easier to analyze than conventional ciphers. Yet another objective is to provide block cipher constructions which will support new approaches in dynamic keying and authentication, as well as other features which can only be practical within the context of a variable size block cipher.

Fundamental Speed Advantage

Ciphering is an overhead cost to communications, and so can never occur fast enough. The ciphers of the present application are based on constructions which produce effective ciphers out of relatively few particularly fast and simple layers. This unusual construction is faster than other techniques; in this sense it succeeds where others have failed.

Fundamental Strength Advantage

Variable size block cipher designs typically make heavy use of keyed substitution tables. The byte-width tables used in the examples of this application are much larger and more complex than the simple fixed tables used in DES. Nevertheless, these larger tables are well within the range of modern resources, and are not unknown in the prior art.

In these variable size block cipher examples, each table will be initialized by shuffling under the control of a key. Since there are 256 factorial (256 x 255 x ... x 2) possible shufflings of a byte-width table, we can reasonably expect that each table in a real system will differ. In contrast, DES uses the same known tables all the time.

When we increase the block size in a variable size block cipher, we typically increase the number of substitution operations; this makes the cipher transformation more complex, even when tables are re-used. We also have the option of using more tables, which will further increase the complexity of the transformation. All things being equal, a larger layer and more tables must inherently make a stronger cipher.

Conventional block ciphers are often designed with "rounds," which repeat a certain core of operations a suggested number of times. Additional rounds can be used in these ciphers. But even the additional rounds still perform the same operations as earlier rounds, and if they use tables, they use the same tables, so it is not at all clear that more rounds necessarily make a stronger cipher. In contrast, added layers in the style of this invention could and should include new substitution tables or a different ordering, and so can be expected to add new strength.

If we equate a layer each of confusion and diffusion in a variable size block cipher with a round in a conventional cipher, the variable size block cipher needs fewer rounds (4.5 versus 8). In this sense, the variable size block cipher succeeds where others have failed.

Better Fit To Existing Systems

Some existing digital systems may have unusual block sizes which a variable size block cipher could better service. In some cases, a designer could choose to accumulate or buffer data until there is enough to encipher as a conventional block. But this buffering could contribute to excessive overall delay, especially with respect to "bursty" data. For example, if only a partial block remains and there is no more data to send, the buffering logic may wait until more data become available before the last data are sent. Presumably, additional complexity could solve this problem, but it might be better to simply avoid the entire issue with variable size block technology.

Buffering is also an overhead operation which must inevitably reduce processing speed in a general-purpose computer realization. The problem of matching a cipher to existing or optimal data block sizes so as to better fit a particular application is a problem with which the literature does not seem particularly concerned.

The speed and other advantages of a variable size block cipher could be advantageous even when used with the fixed-size 64-bit data blocks used by DES and other block ciphers.

Better Fit to Inherently Variable-Size Devices

Modern voice ciphers use a "codec" (COder, DECoder) to convert voice to a digital stream of data. Such a device probably would not produce exactly the block size desired by a conventional block cipher and might ideally produce variable-size data blocks, making a variable size block cipher the best match.

A variable size block cipher would also be a better fit to cipher a database containing fields of various size. Some fields might even be of variable size, again making a variable size block cipher an ideal match, and a fixed-size block cipher a problem.

Speed Advantage From Large Blocks

Where the number of blocks can be reduced by using larger blocks, the total amount of per-block overhead may be reduced from that of a fixed size block cipher.

Strength Advantage From Large Blocks

Block size plays a pivotal role in block cipher strength. For example, an 8-bit substitution table (with 256 entries) might be described as a sort of block cipher, but would be weak as soon as any table entry was used more than once.

Block ciphers attempt to emulate a substitution table of huge size, so that ciphering transformations are very rarely re-used. But a large block can contain more characters, with larger amounts of language, having more variability and even less likelihood of repeating. Thus, the simple ability to cipher large blocks can mean a stronger cipher.

Unexpectedly Supports A New Form of Keying

One way to dynamically key a block cipher is to include the key in the block along with the data. Of course, if we are constrained to use a small, fixed-size block, placing a key in each block will inconveniently reduce the room available for data. In fact, if we want to use an 80-bit key, the key itself will not even fit inside the conventional 64-bit block of DES, let alone leave room for message data. But if we can use a variable size block, the block size can be expanded to include the key plus the original amount of data. That is, this form of keying is really only useful when we have a variable size block cipher.

The advantage of this form of keying is the ability to change the key on a block-by-block basis, with almost no computational overhead. This does expand the block size, but since we can use huge blocks, the amount of expansion can be extremely modest. For example, an 80-bit key is only five percent of a 200-byte block.

Moreover, if we treat the keying value as an authentication code, the extra keying space may be free, since some sort of authentication is probably necessary anyway.

This new form of keying is totally unexpected because the approach is simply impractical with a conventional fixed-size block.

Unexpectedly Supports a Secure Authentication Code

A surprising new way to provide an authentication code is to include an authentication value in the block along with the data. This automatically creates a secure error-check and message authentication code (MAC) in each and every ciphered block.

Of course, if we are constrained to use a small, fixed-size block, placing an authentication value in each block will inconveniently reduce the space available for data. But if we can use a variable size block, the block size can be expanded to include the authentication value plus the original amount of data. That is, this form of authentication is really only useful when we have a variable size block cipher.

When we can use the authentication capability of the variable size block cipher itself, we may be able to eliminate a separate authentication computation. This would almost certainly further increase effective processing speed over systems using conventional fixed-size block ciphers and separate authentication.

With a variable size block cipher, a single block might even hold an entire message. In this case, ciphering and authentication could occur in a single step, without external authentication computation, and without additional overhead.

This type of authentication might even provide a better theoretical basis for generating a Message Authentication Code than the seemingly ad-hoc approaches which are now in use.

This new form of authentication is totally unexpected because the approach is simply impractical with a conventional fixed-size block.

Supports Other Forms of Dynamic Keying

Another way to dynamically key a variable size block cipher is to simply add a layer which in some way combines a key value with the data. For example, a layer of byte-wide exclusive-OR operations could combine the data block with a data-size key. This is not unknown in the prior art. But to the extent that dynamic keying is important in convincing designers that the variable size block ciphers of this application are practical, this is another way to accomplish dynamic keying.

To support variable size blocks, the keying layer could combine data with values from a very simple and fast random number generator--very like a simple stream cipher. With respect to the overall block cipher, a keying layer could be a confusion layer or perhaps a combined confusion and diffusion layer.

Inherently Supports Testable Small Versions

A variable size block cipher can be tested with a small block size, and then exactly the same program or exactly the same hardware (with only a different block size value) used as the production cipher. The advantage here is the ability to test not just a simplified experimental package (which may not contain an error present in the larger cipher), or even a small prototype version, but the actual live cipher itself.

The ability to test the same cipher using small blocks may make it possible to exhaustively test all aspects of that version. This information can be used to refine the design, and extrapolate and estimate the complexity of the large version, which will be far too large to exhaustively test. This solves a problem with construction validation which has apparently never before been recognized or solved in the literature.

Inherently Supports Easier Analysis

The fundamental structure of the variable size block cipher examples in this application all use mathematically-simple yet substantial operations. These operations can be described mathematically, and can be well understood.

For example, all of the substitution tables in these variable size block ciphers are just different arrangements or permutations of the same values. Among other things, this means that we can guarantee that any input change must select a different output value, and thus produce an output change. More general confusion functions, such as those supported in Feistel architectures, do not have this property and do not support this sort of reasoning.

Similarly, the use of layers which provide diffusion across the width of the block in the variable size block ciphers of this application simplifies diffusion questions compared to other ciphers. For example, most conventional Feistel designs apparently find the number of rounds needed to for complete diffusion by experiment. In the variable size block cipher examples given here, we can do the same sort of experiments, but also can reason about the effect of each separate layer, and there are relatively few layers to analyze.

Since there is no way to measure or guarantee the strength of any practical cipher, any improvement in the ability to reason about the mechanism is a big advantage.

Need Not Expand Data At All

The usual implication of collecting variable-length data into fixed-size blocks is that the last block will only be partially filled, but must nevertheless be fully enciphered. This means that a fixed size block cipher generally must be allowed to expand the data by almost a full block during ciphering. Expanded data means taking more storage space and transmission time than would otherwise be needed. A true variable size block cipher at least has a possibility of avoiding any data expansion at all.

Further objects and advantages of my invention will become apparent from a consideration of the drawings and ensuing description.

Summary of the Invention

The present invention uses simple invertible diffusion and confusion layers (or combined layers) to produce a variable size block cipher. Since each of the layers is separately invertible, the cipher itself is invertible, so that enciphered data may be deciphered. Deciphering occurs by producing the inverse of each layer and processing each layer in reverse order from that used in enciphering.

Layers

The layers in the present invention are special in that they use simple operations which operate on small units of data or data elements (for example, bytes). Such layers support extreme speed in processing and are easily realized in hardware or a general purpose computing machine.

Each confusion layer of the present invention is not too different from a weak stream cipher. But because the diffusion layers have no strength, they would be very unusual as a stream cipher. It is only when these layers are used in combination that they unexpectedly provide the overall diffusion and strength needed in a block cipher.

Arbitrary Size

Layers based on small operations are easy to extend to virtually arbitrary size (in steps of the operation size). And if all the layers in the cipher are easily extended, the cipher itself can handle data blocks of arbitrary size (typically in byte-size steps). This yields a variable size block cipher with advantages not available in fixed-size designs.

Some of the advantages of variable size block ciphers include: higher speed, a better fit to existing systems, support for testable small versions, and easier analysis. Moreover, a variable size block cipher unexpectedly supports exciting new methods of dynamic keying and authentication.

Keying

The confusion layers provide primary keying. In a typical confusion layer, each element of the data block is transformed by an associated substitution operation. In a layer of byte-wide invertible substitutions, each table can differ only in the ordering of the values in the table, but there are 256 factorial such orderings. Selecting among all possible table orderings would seem to require a big key, but we can support this by using a key of reasonable size to initialize the state of a random number generator. The sequence of values from the random number generator mechanism can then be used to shuffle all of the tables in the cipher. Both the design of random number generators and their use to shuffle invertible substitutions are well known and are not improved in this application.

Some dynamic keying may be provided by diffusion-layer inputs (these are the start of the diffusion channels on one or the other side of the block). More dynamic keying can be added by including a keying layer in the cipher. This layer would simply combine a key of the block width with the data. Combining could be additive, Latin square, or other cryptographic combining. A key of any effective width can be provided by a keyed random number generator.

Additional dynamic keying can be provided by including a key as part of the data block. This would produce more ciphertext than message (enciphering would appear to expand the data), but the resulting key would be reproduced by deciphering, and could be used to authenticate the original data. Since, in a variable size block cipher, the data block can be as large as desired, even an 80-bit dynamic key could be a minor overhead. And it might well provide an authentication which may be necessary anyway.

Overall Diffusion

Appropriate diffusion layers have been found which support the overall diffusion characteristic needed in a block cipher. The diffusion available in a single fast and simple layer is typically one-way. That is, any change in input value is propagated across the block in just one direction. Accordingly, it is often necessary to use multiple diffusion layers to achieve the overall diffusion required in a block cipher.

Overall diffusion can be produced by having separate layers diffusing in opposite directions. Alternately, overall diffusion can be produced by layers diffusing in the same direction, provided the carry output from an early layer is communicated to the initial value input of one or more later layers. The latter approach would have to be taken carefully, however, since it could circumvent the isolation of intermediate confusion layers, and might be a weakness which could be attacked.

The overall diffusion characteristics for a block cipher are relatively easy to check by experiment: For a random key we choose a random data block and encipher it. Then we change a bit in the original block, encipher that, and compare the result to the original ciphertext, counting the number of bits which changed. We do this for a large number of random blocks and a large number of keys, producing a distribution which is the number of occurrences of each possible number of bit-changes. We know what to expect because the probability of finding any particular number of bit changes c in a block of b bits is:

$$\text{Prob}(\ c\ ) = \frac{\binom{b}{c}}{2^{b}}$$

With a reasonable number of experiments (e.g., 160,000 encipherings of 200-byte blocks) large blocks produce an especially clear and distinct avalanche signature. This is because large blocks have a relatively small range of values for expected bit changes c, and even a huge number of experiments should produce few unexpected values. Overall diffusion tests of the preferred embodiment examples in Figs. 1 through 4 have shown particularly good results, sometimes surprisingly better than very similar architectures.

Brief Description of the Drawings

FIG. 1 is a data-flow diagram of a fast and effective 80-bit block cipher which uses keyed byte-wide substitution operations for confusion and byte-wide exclusive-OR operations for diffusion.

FIG. 2 is a data-flow diagram of an 80-bit block cipher similar to Fig. 1, but with improved diffusion and strength.

FIG. 3 is a data-flow diagram of an 80-bit block cipher which uses byte-wide Balanced Block Mixer operations for diffusion.

FIG. 4 is a data flow diagram of an 80-bit block cipher which uses Latin square combining operations for simultaneous confusion and diffusion.

Description of The Presently Preferred Exemplary Embodiments

Figures 1 through 4 show 80-bit block cipher designs. The 80-bit block was chosen as being larger than the 64-bit block of DES, and not a power-of-2 size (as many ciphering architectures must be). Since each design uses exactly the same block size, their effectiveness can be compared directly.

Figures 1 through 4 illustrate ciphers composed of horizontal rows or layers of similar components, and vertical columns which often contain different component types. The heart of each cipher is the vertical column composed only of operations which are one element in width. In these particular designs, each element is one byte wide; this means that all operations are byte-width operations, and all internal data paths are eight bits wide.

Each column in any one of these ciphers contains the same connections and operations. The columns differ only in the particular adjacent columns (if any) to which their data and diffusion inputs and outputs are connected. Each cipher can be extended to arbitrary size by adding more columns. We choose to discuss the leftmost column only, without loss of generality, since the structure of each column is the same except for the connections which transport data beyond that column.

Since diffusion is necessary across the width of the block, each column communicates with adjacent columns. This means that the leftmost and rightmost columns will produce diffusion carry outputs (which can be ignored) and will require external input values.

Figures 1 through 4 are data-flow diagrams. That is, the figures represent the flow of data from some source through a data path to an operation, the operation itself, and the flow of the result through another data path to some data sink or user of the result value. In a hardware realization, a data path might be actual metallic connections. In a general-purpose computer realization, a data path might simply represent the movement of data from one operation to another.

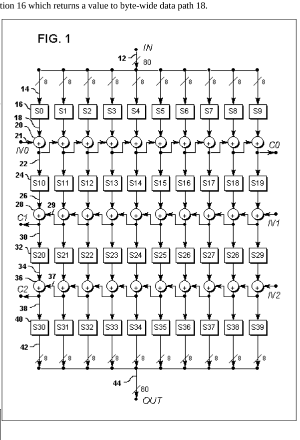
In practice, processing might occur sequentially, especially in a general-purpose computer. In this case, each layer might be processed separately, one element at a time. Depending on the particular implementation involved, it could be more efficient to combine several layers into a single pass across the block, and other layers into another pass, generally in the opposite direction.

Figures 1 through 4 illustrate practical ciphers using 10 element positions or columns of 8-bit operations. Since similar operations are used in each element position, it is clear that element positions could be replicated as necessary to produce any reasonable block size. Most of these designs have been tested both with 80-bit (10-byte) and 1600-bit (200-byte) blocks.

Fig. 1--Simple Exclusive-OR Diffusion

Fig. 1 is an example of an 80-bit block cipher built solely from variable size layers. This makes the cipher easily extendible (in byte-by-byte steps) to arbitrary size, either at design-time, or dynamically during operation.

An 80-bit input block is conducted into the cipher mechanism by block-wide data path 12. Block-wide data path 12 is split into multiple data paths each of element size, including byte-wide data path 14. In the specific example of Fig. 1, the element size is a byte. Data path 14 conducts data to byte-wide substitution operation 16 which returns a value to byte-wide data path 18.



Data path 18 conducts its data to byte-wide exclusive-OR operation 20. Exclusive-OR 20 additively combines the data from data path 18 and byte-wide data path 21 from external value IV0 and returns the result to data path 22. Data path 21 carries right-going diffusion from the left-adjacent column; in the leftmost column this is byte value IV0 (Initial Value 0) from outside the cipher. Data path 22 continues the right-going diffusion to the right-adjacent column; in the rightmost column this is reported out of the cipher as byte value C0. The carry outputs and IV inputs illustrate that columns could easily be added to either side of the existing design.

Data path 22 also conducts its data to byte-wide substitution operation 24, which returns a value to byte-wide data path 26. Substitutions 16 and 24 and all the other substitutions in the cipher represent separate data processing operations, but are not necessarily different tables. The actual table used at any particular substitution position might be chosen from many such tables, depending upon element position, layer, some data value, or a combination of these.

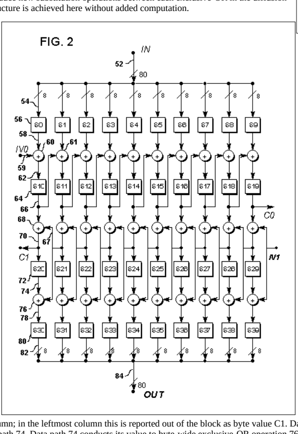
Data path 26 conducts its data to byte-wide exclusive-OR operation 28. Exclusive-OR 28 additively combines the data from data path 26 and byte-wide data path 29, and returns the result to data path 30. Data path 29 carries left-going diffusion from the right-adjacent column; in the rightmost column this is byte value IV1 from outside the cipher. Data path 30 continues the left-going diffusion to the left-adjacent column; in the leftmost column this carry is reported out of the cipher as byte value C1.

Data path 30 conducts data to byte-wide substitution table 32, which returns a result to data path 34, which conducts its data to byte-wide exclusive-OR 36. Exclusive-OR 36 additively combines the data from data path 34 and byte-wide data path 37, and returns the result to byte-wide data path 38. Data path 37 carries left-going diffusion from the right-adjacent column; in the rightmost column this is byte value IV2 from outside the cipher. Data path 38 continues the left-going diffusion to the left-adjacent column; in the leftmost column, this is reported out of the cipher as byte value C2. Data path 38 also conducts its value to byte-wide substitution table 40, which returns a result to data path 42. Data path 42 is simply collected along with the results from the other columns, to form block-wide data path 44, which conducts the result out of the cipher mechanism.

Fig. 2--Diffusion Across Layers

The 80-bit block cipher example of Fig. 2 is much like the similar structure of Fig. 1, with improvements in diffusion. Like Fig. 1, Fig. 2 includes, in alternation, four layers of substitution operations and three layers of exclusive-OR operations. Like all of these variable size block cipher designs, the structure of each column is the same, although the actual substitution tables used will vary.

Unlike Fig. 1, some of the diffusion paths used in Fig. 2 include intermediate substitutions using substitutions already in place. Clearly, it would be possible to introduce new substitution operations between each exclusive-OR in the diffusion paths, but much of the benefit of such a structure is achieved here without added computation.



An external data block is conducted to the cipher mechanism of Fig. 2 through block-wide data path 52. Eight bits of block 52 are conducted by byte-wide data path 54 to byte-wide substitution operation 56. Substitution 56 produces a result on byte-wide data path 58, which conducts that value to byte-wide exclusive-OR operation 60. Exclusive-OR 60 additively combines the value on data path 58 with the value on byte-wide data path 59, and produces a result on byte-wide data path 62. Data path 59 carries right-going diffusion from the left-adjacent column; in the leftmost column, this is byte value IV0 (Initial Value 0) from outside the cipher. Data path 62 conducts the combined value to byte-wide substitution operation 64, which produces a result on byte-wide data path 66. Data path 66 continues the right-going diffusion to the right-adjacent column; in the rightmost column this is reported out of the cipher as byte value C0 (carry 0). The carry outputs and IV inputs illustrate that columns could easily be added to either side of the existing design.

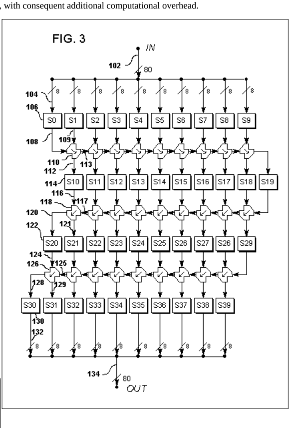
The value on data path 66 is conducted to byte-wide exclusive-OR operation 68, which additively combines the value on data path 66 with the value on byte-wide data path 67 and produces a result on byte-wide data path 70. Data path 67 carries left-going diffusion from the right-adjacent column; in this rightmost column this is byte value IV1 from outside the cipher. Data path 70 continues left-going diffusion to the left-adjacent column; in the leftmost column this is reported out of the block as byte value C1. Data path 70 also conducts its value to byte-wide substitution operation 72, which produces a result on byte-wide data path 74. Data path 74 conducts its value to byte-wide exclusive-OR operation 76, which additively combines the value from data path 74 and byte-wide data path 67 and produces a result on byte-wide data path 78. Data path 67 carries left-going diffusion from the right-adjacent column. Data path 78 conducts its value to byte-wide substitution 80, which produces a result on byte-wide data path 82. Data path 82 is simply collected with the other byte-wide data paths in the same layer, producing block-wide data path 84 which conducts the final result out of the cipher.

Fig. 3--Balanced Block Mixing for Diffusion

The example 80-bit block cipher of Fig. 3 exploits the reversible two-port nature of Balanced Block Mixing to provide somewhat better diffusion than exclusive-OR, and to avoid the need for external Initial Values.

Because the structure of the mixing component differs from the exclusive-OR used in Fig. 1 and Fig. 2, the diffusion architecture of Fig. 3 is accordingly different. In particular, there is always one less Balanced Block Mixing in the diffusion layers than there are element-wide data paths and substitution elements in the confusion layers. We can think of the Balanced Block Mixing as fitting in the space between adjacent confusion columns.

Again, it would be possible to introduce new substitution operations in the diffusion paths between Balanced Block Mixings, with consequent additional computational overhead.

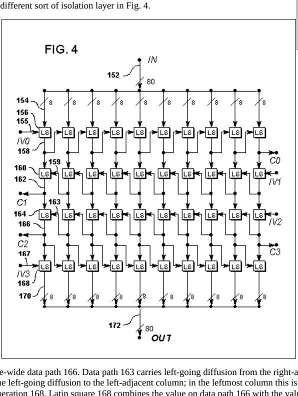


An external data block is conducted into the cipher on block-wide data path 102 and eight bits of that block are conducted by byte-wide data path 104 to byte-wide substitution operation 106. Substitution 106 produces a result on byte-wide data path 108 which is conducted to one of the input ports of byte-wide Balanced Block Mixer 110. Mixer 110 combines the values on data path 108 with the value on byte-wide data path 109 and produces result values on byte-wide data paths 112 and 113. Data path 109 carries the confusion value from top confusion layer in the right-adjacent column. Data path 113 carries right-going diffusion to the right-adjacent mixer or, in the rightmost column, the rightmost confusion element of the next confusion layer. Data path 112 carries its value to byte-wide substitution operation 114, which produces a result on byte-wide data path 116 which conducts that value to one input port of Balanced Block Mixer 118. Mixer 118 combines the value on data path 116 and byte-wide data path 117 and produces result values on byte-wide data paths 120 and 121. Data path 117 carries left-going diffusion from the right-adjacent mixer; in the leftmost column this is data path 120 and is connected to the leftmost confusion element of the next confusion layer. Data path 121 carries diffusion to the substitution which corresponds to substitution 122 in the right-adjacent column. Data path 120 carries its value to byte-wide substitution operation 122, which produces a result on byte-wide data path 124 which is conducted to one input port of Balanced Block Mixer 126. Mixer 126 combines the value on data path 124 with the value on byte-wide data path 125, and produces result values on byte-wide data paths 128 and 129. Data path 125 carries left-going diffusion from the right-adjacent mixer. In the rightmost column, the data path corresponding to path 125 comes from the rightmost element of the previous confusion layer. In the leftmost column, the data path corresponding to path 125 is connected to the leftmost confusion element of the next confusion layer. Data path 129 carries diffusion to the substitution which corresponds to substitution 130 in the right-adjacent column. Data path 128 carries its value to byte-wide substitution operation 130, which produces a result on byte-wide data path 132. Data path 132 is collected with the result values from the other substitutions in the same layer as substitution 130 into block-wide data path 134. Data path 134 then conducts the result out of the cipher for external use.

Fig. 4--Latin Square Combined Confusion and Diffusion

The 80-bit block cipher of Fig. 4 is unique in that, of the four examples, it creates a cipher from a single component type. That component type is the Latin square combiner, which simultaneously provides both confusion and diffusion. In a byte-width Latin square, each of 256 rows and 256 columns will have exactly one occurrence of each byte value 0..255. Before operation, the needed Latin square (or squares) are constructed under the control of one or more cryptographic keys. Each operation in this example might well use the same keyed Latin square.

We can directly compare Fig. 4 to Fig. 1, if we consider a confusion layer and diffusion layer in Fig. 1 as a single layer in Fig. 4. In both cases, we first have one confusion and diffusion to the right, and two to the left. We are then left with a single confusion layer at the bottom of Fig. 1, and a different sort of isolation layer in Fig. 4.



An external data block is conducted into the cipher on block-wide data path 152 and eight bits of that block are conducted by byte-wide data path 154 to byte-wide Latin square combining operation 156. Latin square 156 combines the value on data path 154 with the value on byte-wide data path 155 and produces a result value on byte-wide data path 158. Data path 155 carries right-going diffusion from the left-adjacent column; in the leftmost column, this is byte value IV0 (Initial Value 0) from outside the cipher. Data path 158 continues the right-going diffusion to the right-adjacent column; in the rightmost column this is reported out of the cipher as byte value C0 (carry 0). Data path 158 also conducts its value to Latin square combining operation 160. Latin square 160 combines the value on data path 158 with the value on byte-wide data path 159 and produces a result value on byte-wide data path 162. Data path 159 carries left-going diffusion from the right-adjacent column; in the rightmost column this would be byte value IV1 from outside the cipher. Data path 162 continues the left-going diffusion to the left-adjacent column; in the leftmost column this is reported out of the cipher as byte value C1. Data path 162 also carries its value to byte-wide Latin square combining operation 164. Latin square 164 combines the value on data path 162 with the value on byte-wide data path 163 and places a result value on byte-wide data path 166. Data path 163 carries left-going diffusion from the right-adjacent column; in the rightmost column this would be byte value IV2 from outside the cipher. Data path 166 continues the left-going diffusion to the left-adjacent column; in the leftmost column this is reported out of the cipher as byte value C2. Data path 166 also carries its value to byte-wide Latin square combining operation 168. Latin square 168 combines the value on data path 166 with the value on data path 167 and places a result value on byte-wide data path 170. Data path 167 carries an isolation value from the left-adjacent column; in the leftmost column this is byte value IV3 from outside the cipher. Data path 166 provides isolation to the right-adjacent column; in the rightmost column this would be reported out of the cipher as byte value C3. Data path 170 is collected with the result values from the other Latin squares in the same layer as Latin square 168 into block-wide data path 172. Data path 172 then conducts the result out of the cipher for external use.

Operation

Before operation, all the various substitution tables to be used are initialized and shuffled under the control of a key, and any Initial Values will also be selected. In general, each table will be a unique ordering of the same values.

When substitution operations are indicated, the particular table used at each position could be fixed at design time, or selected dynamically, during operation, from among many such tables.

In all cases, deciphering is performed by very similar constructions which process data from the bottom to the top. In most cases, the diagram for deciphering is exactly the same as that for enciphering, with all downward data paths replaced by upward data paths.

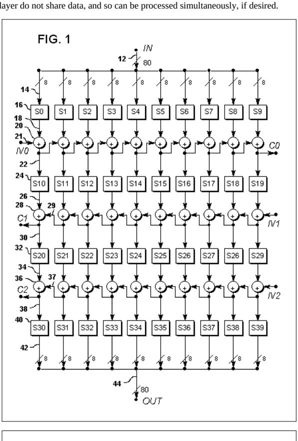
Normally, with exclusive-OR or Latin square diffusion, the direction of diffusion does not change when deciphering. However, when Balanced Block Mixers are used, for each mixer, both outputs become inputs, and both inputs become outputs, meaning that the diffusion direction is also reversed.

When deciphering, each substitution table is replaced by its inverse, and each Latin square is replaced by its appropriate inverse. If the Balanced Block Mixers are keyed, each of those will be replaced by its inverse; like exclusive-OR, the usual Balanced Block Mixer is its own inverse.

Fig. 1--Simple Exclusive-OR Diffusion

In operation, an external 80-bit block to be enciphered is applied to block-wide data path 12, and external byte values (probably developed from a key) are applied to IV0, IV1, and IV2. The cipher then splits the data block into ten element-wide values for processing in separate columns.

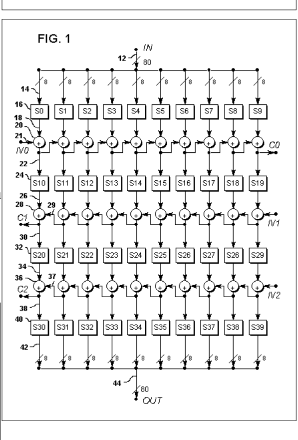
The first layer is a variable size confusion layer which includes substitution 16. The ten substitution operations in the first layer do not share data, and so can be processed simultaneously, if desired.



The second layer is a variable size diffusion layer which includes exclusive-OR 20. Here, the ten exclusive-OR operations, starting at IV0 and ending at output C0, do share data for diffusion. This implies a left-to-right ordered sequence of operations.

The third layer is another confusion layer, and includes byte-wide substitution operation 24. This layer functions just like the previous confusion layer.

To understand the purpose of the top three layers, it is useful to consider the operation in the case of two different input blocks: first, some random block, and second, the same block with only the leftmost bit changed. The different or changed bit produces a different value on data path 14, which produces a different code from substitution 16. At least one bit will change on byte-wide data path 18, and will be propagated across the full width of the block through the diffusion layer which includes exclusive-OR 20. This will change the value into each of the substitutions in the confusion layer which includes substitution 24, thus changing the result from each substitution. In this way, a change in any one input bit affects the column which includes that bit, and also each column to the right. The top three layers might be computed in one sequential pass.



The next diffusion layer, which includes exclusive-OR 28, operates like the previous diffusion layer, but in the opposite direction. This means that we cannot do all of the operations in one column before doing some of the operations in other columns. That is, the diffusion carry from adjacent columns generally constrains the processing to occur by layer (or groups of layers), instead of by column.

The next confusion layer, which includes substitution 32, operates like the previous confusion layer. The next diffusion layer, which includes exclusive-OR 36, operates like the previous diffusion layer. And the next confusion layer, which includes byte-wide substitution operation 40, operates like the previous confusion layer. The bottom four layers, which include two diffusions, but in only one direction, might be computed in a second sequential pass.

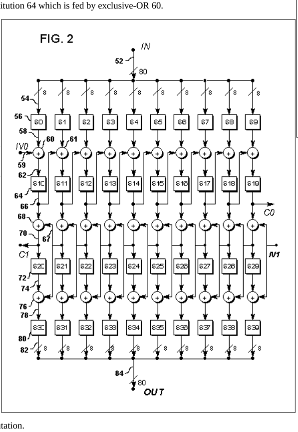
The resulting values from each column of the last layer are simply collected as the output.

To decipher the block, note that each layer is separately invertible. We simply produce the inverse tables of those used for enciphering, and process the block "up," from OUT to IN in Fig. 1. That is, a data-flow diagram for deciphering is exactly the same as Fig. 1 with only the down-arrows changed to up-arrows. The direction of diffusion in each diffusion layer remains unchanged.

This technology supports the ciphering of blocks of various size, including huge blocks. Moreover, the total computational effort required is on the order of four table look-ups per byte ciphered. This is probably an order of magnitude less computation than that required by DES.

Fig. 2--Diffusion across Layers

Fig. 2 is essentially the same as Fig. 1, with improved diffusion. It is found by experiment that diffusion improves when a diffusion layer includes substitution operations between exclusive-OR's. To avoid additional operations or components, the diffusion path is taken from the result of the next confusion layer. In particular, exclusive-OR 61, in the same layer as exclusive-OR 60, uses the result from substitution 64 which is fed by exclusive-OR 60.



The layer including exclusive-OR 68 is a conventional left-diffusion layer. But the layer including exclusive-OR 76 takes its diffusion inputs from the layer including exclusive-OR 68. This construction is thought to improve the strength of the cipher, by limiting the ability to isolate a single table.

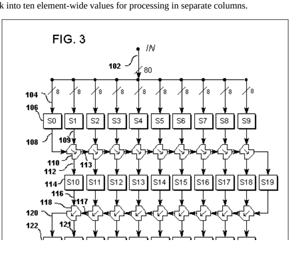
When advantageous, a general-purpose computer realization might scan the block only twice: One scan would occur from left to right for the top three layers, which diffuse to the right. The other scan would be for the bottom four layers, which diffuse to the left. But in some cases it could be more efficient to simply scan each layer in sequence.

To decipher the block, again note that each layer is separately invertible. We simply produce the inverse tables of those used for enciphering, and process the block "up," from OUT to IN in Fig. 2. That is, a data-flow diagram for deciphering is exactly the same as Fig. 2 with only the down-arrows changed to up-arrows. The direction of diffusion in each diffusion layer remains unchanged.

Fig. 3--Balanced Block Mixing for Diffusion

Fig. 3 differs from Fig. 1 in its diffusion layers. The use of byte-width Balanced Block Mixing operations produces a slightly different layer structure. This provides somewhat better diffusion than Fig. 1, but also requires a little more computation.

In operation, an external 80-bit block to be enciphered is applied to block-wide data path 102. The cipher splits the data block into ten element-wide values for processing in separate columns.



The first layer is a confusion layer which includes substitution 106. The ten substitution operations in the first layer do not share data, and so can be processed simultaneously, if desired.

The second layer is a Balanced Block Mixer diffusion layer which includes mixer 110 and diffuses to the right. Here, the nine Balanced Block Mixers interleave the confusion columns. A mixer operates between each pair of adjacent confusion columns. The mixers do share data, which implies a left-to-right ordered sequence of operations.

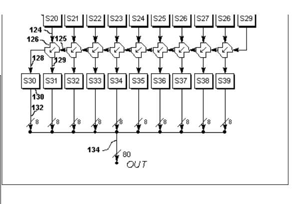
The next five layers alternate byte-width confusion and Balanced Block Mixer layers which diffuse to the left. Again, it may be reasonable to do the top three layers in one pass, and the bottom four layers in another pass, where this is more efficient than layer-by-layer scans.

The resulting values from each column of the last layer are simply collected as the output.

To decipher the block, note that each layer is separately invertible. We simply produce the inverse confusion tables of those used for enciphering, and process the block "up," or from OUT to IN in Fig. 3. That is, a data-flow diagram for deciphering is the same as Fig. 3 with all arrows reversed. The Balanced Block Mixer outputs are made inputs, and the inputs outputs, and diffusion occurs in the opposite direction.

Fig. 4--Latin Square Combined Confusion and Diffusion

In Fig. 4, the exclusive use of Latin square combining provides simultaneous confusion and diffusion in each layer. Here,



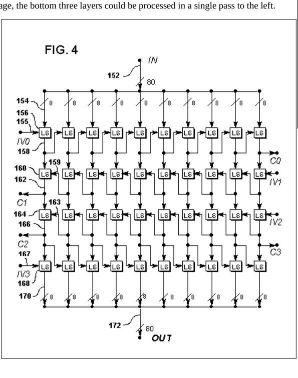
since each indicated operation is part of a diffusion, there is an implicit ordering of all operations corresponding to the diffusion direction in each row. For example, in a general-purpose computer realization, one might cause the machine to scan across the top row of operations from left to right. Then one might cause the machine to perform the middle two rows of operations in one or two passes from right to left. Then the machine could perform the bottom row of operations. Or, perhaps, with some additional temporary storage, the bottom three layers could be processed in a single pass to the left.

As usual, the cipher splits the block-wide data block into ten element-wide values for processing as separate, but communicating, columns. Each column is the same, so the cipher as a whole can be expanded element-by-element to arbitrary size.

The top layer of operations simultaneously confuse the input element values and diffuse to the right. The next two layers operate similarly, to the left. The bottom layer serves to isolate the result values.

To decipher the block, note that each layer is separately invertible. We simply produce the appropriate "inverse" square from that used for enciphering, and process the block "up," from OUT to IN in Fig. 4. That is, a data-flow diagram for deciphering is exactly the same as Fig. 4 with only the down-arrows changed to up-arrows. The direction of diffusion in each diffusion layer remains unchanged.

FIG. 4



Glossary

Additive Combiner

An additive combiner mechanism realizes numerical concepts similar to addition.

One example is byte addition modulo 256, which simply adds two byte values, each in the range 0..255, and produces the remainder after division by 256, again a value in the byte range of 0..255. Because subtraction is closely related to addition, byte subtraction mod 256 is also an additive combiner.

Another example is the bit-level exclusive-OR operation, which simply adds two bit values, each in the range 0..1, and produces the remainder after division by 2. Thus, exclusive-OR is also addition mod 2, and a byte-level exclusive-OR operation is a polynomial addition of mod 2 elements.

ASCII

A particular coding or correspondence between text characters and numeric values. ASCII is an acronym for American Standard Code for Information Interchange.

Authentication

One of the objectives of cryptography: Assurance that a message is genuine; assurance that a message has not been modified in transit or storage.

Authenticating Block Cipher

A block cipher mechanism which inherently contains an authentication value or field.

An authenticating block cipher can be created by collecting an arbitrary authentication or key value in the data block along with the plaintext data. Of course this is only practical if there is enough room for a reasonable size key and a reasonable amount of plaintext data. But there will be plenty of room if the cipher is a variable size block cipher.

Avalanche

The observed property of a block cipher constructed in layers or "rounds" with respect to a tiny change in the input. The change of a single input bit generally produces multiple bit-changes after one round, many more bit-changes after another round, until, eventually, about half of the block will change. An analogy is drawn to an avalanche in snow, where a small initial effect can lead to a dramatic result. See overall diffusion.

Balanced Block Mixer

A process for balanced block mixing or any implementation (for example, hardware, computer software, hybrids, or the like) for performing balanced block mixing. See balanced block mixing.

Balanced Block Mixing

A mechanism as described in my pending application 08/380,960 titled Cipher Mechanisms with Fencing and Balanced Block Mixing filed 1995 Jan 30. In the present application, only a two-input-port two-output-port version is considered. A Balanced Block Mixer is a two-input-port two-output-port mechanism with various properties:

1. The mapping is one-to-one and invertible: Every possible input value (including both ports) to the mixer produces a different output value, and every possible output value is produced by a different input value;
2. Each output port is a function of all input ports;
3. Any change to any one of the input ports will produce a change to every output port;
4. Stepping any one input port through all possible values (while keeping the other inputs fixed) will step every output port through all possible values.

If the two input ports are labeled A and B, the two output ports are labeled X and Y, and we have some irreducible mod 2 polynomial p of degree appropriate to the port size, a Balanced Block Mixer is formed by the equations:

```
X = 3A + 2B (mod 2)(mod p),

Y = 2A + 3B (mod 2)(mod p).
```

This particular Balanced Block Mixer is a self-inverse, and so can be used without change whether enciphering or deciphering. One possible value for p is 100011011 in binary (this is a mod 2 polynomial).

Balanced Combiner

In the context of cryptography, a combiner mixes two input values into a result value. A balanced combiner must provide a balanced relationship between each input and the result.

In the usual balanced combiner, any particular result can be produced by any value on one input, simply by selecting some appropriate value for the other input. In this way, knowledge of only the output value provides no information about either input.

The common examples of cryptographic combiner, including byte exclusive-OR (mod 2 polynomial addition), byte addition (integer addition mod 256), or other "additive" combining, are perfectly balanced. Unfortunately, these simple combiners are also very weak, being inherently linear and without internal state.

In contrast, there do exist nonlinear combiners with internal state, for example Latin Square combiners. See Latin square combiners.

Bit

The smallest possible unit of information. A Boolean value: True or False; Yes or No; one or zero. All information to be communicated or stored digitally is coded in some way which fundamentally relies on individual bits. Alphabetic characters are often stored in eight bits, which is a byte.

Block

A block (or data block) is some amount of data treated as a single unit. For example, the US Data Encryption Standard (DES) block cipher uses a 64-bit block: it ciphers 64 bits (typically eight ASCII characters) at once. It is not normally possible to block cipher just a single bit or a single byte of a block.

An arbitrary stream of data can always be partitioned into one or more fixed-size blocks, but it is likely that at least one block will not be completely filled. Using fixed-size blocks generally means that the associated system must support data expansion in enciphering, if only by one block. Handling even minimal data expansion may be difficult in some systems. (It may be possible to use a stream cipher on the last partial block, but if the stream cipher were truly satisfactory, we would have no need for the block cipher in the first place.)

Block Cipher

A block cipher is a mechanism which emulates a Simple Substitution of a size which is far too large to realize.

Like a Simple Substitution, a block cipher mechanism can transform a plaintext block into a ciphertext block under control of a key. (Simple Substitution is keyed by shuffling the contents of the substitution table.) A similar or related cipher mechanism can transform a ciphertext block back into plaintext when given the same key.

In an ideal block cipher, changing even a single bit of the input block must change all bits of the ciphertext result with probability 0.5. This means that about half of the bits in the output will change for any different input block, even for differences of just one bit. This is overall diffusion and is present in a block cipher, but not in a stream cipher.

A block cipher must treat all of the data in the block in ways which lead to statistically similar results. Since a block cipher must be built up from smaller components, a major part of the design concerns how every input bit can be effectively and reversibly mixed with every other input bit.

A block cipher differs from a cryptographic hash function, which is not reversible, and so does not, by itself, support deciphering.

Block Size

The amount of data in a block. For example, the size of the DES block is 64 bits or 8 bytes or 8 octets.

Byte

A collection of eight bits. Also called an "octet." A byte can represent 256 different values or symbols. The common 7-bit ASCII codes used to represent characters in computer use are generally stored in a byte; that is, one byte per character.

Cipher

A secrecy mechanism which operates on individual characters or bits independent of semantic content. As opposed to a secret code, which generally operates on words, phrases or sentences, each of which may carry some amount of complete meaning.

Ciphering

The general term which includes both enciphering and deciphering.

Ciphertext

The result of enciphering. Ciphertext will contain the same information as the original plaintext, but hide the original information, typically under the control of a key. Without the key it should be impractical to recover the original information from the ciphertext.

Code

Symbols or values which stand in place of other symbols or values.

Classically, a numeric value might represent a whole word or phrase so as to decrease the cost of telegraph messages. In modern usage, a code is a correspondence between information (such as character symbols) and values (such as the ASCII code). Coding is a very basic part of modern computation and generally implies no secrecy or information hiding. Some codes are "secret codes," however, and then the transformation between the information and the coding is kept secret.

Combiner

In a cryptographic context, a combiner is a mechanism which mixes two data sources into a single result.

A combiner is often used to encipher data, so it is normally necessary that the process be reversible to decipher data. That is, given the result and one of the input values, it is usually necessary that the other input value can be constructed by some related extractor mechanism.

Confusion

Those parts of a cipher mechanism which change the correspondence between input values and output values. In contrast, diffusion propagates changes in one part of a block to other parts of the block.

Confusion Layer

A confusion layer acts to change the value (or values) of the data. When the confusion layer is composed of substitution elements, typically each byte of the local input block is translated through a substitution table.

Confusion layers act to hide and protect the weak diffusion layers, and also to expand the bit-changes propagated by the diffusion layers.

Cryptanalysis

That aspect of cryptology which concerns the strength analysis of a cryptographic system, and the penetration or "breaking" of a cryptographic system.

Cryptographic Mechanism

A process for enciphering and/or deciphering, or an implementation (for example, hardware, computer software, hybrid, or the like) for performing that process.

Cryptography

That aspect of cryptology which is the art and science of secret messages. Cryptography includes secrecy and authentication.

Modern cryptography generally depends upon translating a message into one of an astronomical number of different intermediate representations, as selected by a key. If all possible intermediate representations have similar appearance, it may be necessary to try all possible keys to find the one which deciphers the message. By creating mechanisms with an astronomical number of keys, we can make this approach impractical.

Cryptology

The field of study which generally includes both cryptography and cryptanalysis.

Decipher

The inverse of enciphering. The process which can, if given the proper key, reveal the information or plaintext hidden in ciphertext.

Decryption

The general term for extracting information which was hidden by encryption.

DES

The particular block cipher which is the U.S. Data Encryption Standard. A 64-bit block cipher with a 56-bit key organized as 16 rounds of operations.

Diffusion

Diffusion is the property of an operation such that changing one bit (or byte) of the input will change adjacent or near-by bits (or bytes) after the operation.

Perhaps the best diffusing component is substitution, but this diffuses only within the particular element substituted. Substitution-permutation ciphers move the bits of each substituted element to other elements and substitute again, and do this repeatedly.

Most block cipher designs have fixed, non-expandable diffusion components, and so are effectively limited by these components to a particular data block size.

Also see Overall Diffusion.

Dynamic Keying

That aspect of a cipher which allows a key to be changed with minimal overhead. A dynamically-keyed block cipher might impose little or no additional computation to change a key on a block-by-block basis. The dynamic aspect of keying could be just one of multiple keying mechanisms in the same cipher.

One way to have a dynamic key in a block cipher is to include that key in the block along with the plaintext data. This is normally practical only with blocks of huge size, or variable size blocks.

Another way to have a dynamic key in a block cipher is to add a confusion layer which mixes the key value with the block. For example, exclusive-OR could be used to mix a 64-bit key with a 64-bit data block. This same layer could be made a variable size layer by using some part of the key (some byte) depending on the element position in the data block.

Diffusion Layer

A diffusion layer acts to mix a block. In a variable size block cipher design, diffusion frequently occurs only in one direction in a particular layer, thus implying that multiple diffusion layers would generally be required in a practical cipher.

Diffusion layers are intended to propagate information from one processing position to another. This means that the first such element may require an input value which is not produced by a previous processing position. Moreover, the last element will produce an output "carry" to a position which does not exist. Nevertheless, the carry serves to indicate the extendible nature of a variable size layer.

It is convenient to call the first input value to each diffusion layer an initial value or IV, and the use of some sort of IV is common in block cipher designs. In the variable size block designs of this application, it is usual to require an IV for each confusion layer. The IV inputs can be used to provide two to four bytes of inherent dynamic keying.

Dynamic Substitution Combiner

The combining mechanism described in US. Patent 4,979,832 titled Dynamic Substitution Combiners and Extractors issued Dec. 25, 1990.

Dynamic Substitution is the use of substitution as a combiner, in which the arrangement of the table changes dynamically, under the control of one of the combiner inputs, during operation.

In the usual case, a Simple Substitution table is created and the arrangement of the table shuffled under the control of a key. One combiner input value is used to select a value from within that table to be the combined result. The other combiner input value is used simply to select an entry, and then the values at the two selected entries are exchanged.

Element

In the context of a variable size block cipher, an element is the width of a single column or ciphering unit, often a byte.

Encipher

The process which will transform information or plaintext into one of plethora of intermediate forms or ciphertext, as selected by a key.

Encryption

The general term for hiding information through secret code or cipher.

Extractor

In a cryptographic context, an extractor is a mechanism which produces the inverse effect of a combiner. This allows data to be enciphered in a combiner, and then deciphered in an extractor. Sometimes an extractor is exactly the same as the combiner, as is the case for exclusive-OR.

Fencing Layer

A fencing layer is a variable size block cipher layer composed of small (and therefore realizable) substitutions. Typically the layer contains many separate keyed substitution tables. To make the layer extendible, either the substitutions can be re-used in some order, or in some pre-determined sequence, or the table to be used at each position selected by some computed value.

Intermediate Block

In the context of a layered block cipher, the data values produced by one layer then used by the next.

In some realizations, an intermediate block might be wired connections between layer hardware. In the context of a general purpose computer, an intermediate block might represent the movement of data between operations, or perhaps transient storage in the original block.

Key

The general concept of protecting things with a "lock," thus making those things available only if one has the correct "key."

In cryptography we have various kinds of keys, including a User Key (the key which a user actually remembers), which may be the same as an Alias Key (the key for an alias file which relates correspondent names with their individual keys). We may also have an Individual Key (the key actually used for a particular correspondent); a Message Key (normally a random value which differs for each and every message); a Running Key (the confusion sequence in a stream cipher, normally produced by a random number generator); and perhaps other forms of key as well.

In general, the value of a cryptographic key is used to initialize the state of a cryptographic mechanism.

Keyed Substitution

Two substitution tables of the same size with the same values can differ only in the ordering of the values in the table. But we can key a substitution table by selecting or creating a particular ordering from the key. This is easily accomplished by shuffling the table under the control of a random number generator which is initialized from a key.

Latin Square

A Latin square is a well-known mathematical construct which is a two-dimensional array or matrix of symbols, such that each row and each column contains each symbol exactly once. Consider a Latin square of order 4, which can be represented as a 4 x 4 matrix with four symbols. It turns out that there are exactly 24 such squares, one of which is: ((2,0,1,3), (1,3,0,2), (0,2,3,1), (3,1,2,0)).

Latin Square Combiner

A cryptographic combining mechanism in which one input selects a column and the other input selects a row in an existing Latin square; the value of the selected element is the combiner result.

A Latin square combiner is inherently balanced, because for any particular value of one input, the other input can produce any possible output value. A Latin square can be treated as an array of substitution tables, each of which are invertible, and so can be reversed for use in a suitable extractor. As usual with cryptographic combiners, if we know the output and a specific one of the inputs, we can extract the value of the other input.

For example, a tiny Latin square combiner might combine two 2-bit values each having the range zero to three (0..3). That Latin square would contain four different symbols (0, 1, 2, and 3), and thus be a square of order 4, one of which can be written as:

```
2  0  1  3
1  3  0  2
0  2  3  1
3  1  2  0.
```

With this square we can combine the values 0 and 2 by selecting the top row (row 0) and the third column (column 2) and returning the value 1.

When extracting, we will know a specific one (but only one) of the two input values, and the result value. Suppose we know that row 0 was selected during combining, and that the output was 1: We can check for the value 1 in each column at row 0 and find column 2, but this involves searching through all columns. We can avoid this overhead by creating the row-inverse of the original Latin square (the inverse of each row), in the well-known way we would create the inverse of any invertible substitution. For example, in row 0 of the original square, selection 0 is the value 2, so, in the row-inverse square, selection 2 should be the value 0, and so on:

```
1  2  0  3
2  0  3  1
0  3  1  2
3  1  2  0.
```

Then, knowing we are in row 0, the value 1 is used to select the second column, returning the unknown original value of 2.

A practical Latin square combiner might combine two bytes, and thus be a square of order 256, with 65,536 byte entries. In such a square, each 256-element column and each 256-element row would contain each of the values from 0 through 255 exactly once.

Latin square combining is discussed extensively in my pending application 08/380,960 Cipher Mechanisms with Fencing and Balanced Block Mixing filed 1995 Jan 30.

Layer

In the context of block cipher design, a layer is particular transformation or operation applied across the full width of the block. Layers can be confusion layers (which simply change the block value), diffusion layers (which propagate changes across the block in at least one direction) or both. In some cases it is useful to do multiple operations as a single layer to avoid the need for internal temporary storage blocks.

A layer will generally contain a number of processing positions, typically corresponding to byte positions in the data block. Actual realizations may or may not use a separate processing element for each position; for example, some realizations might re-use a single processing element in each position. Other realizations might select a particular processing element for each position from among many such elements.

The input to the first layer will be an external data block; that layer will then produce a result block which can be seen as an intermediate block, to be conducted to the next layer. However, most layers will neither increase nor decrease the amount of data in the block, and will not need to access any but the previous block value. Therefore, in software realizations, most layers can simply deposit their result back into the external data block storage. In this way, there need exist no extra physical storage for the communication between each layer. The final layer may place its result into the external data block, which then becomes the output block.

In diffusion layers, it often happens that exactly the same operation can be applied at each position. This makes it easy to handle a variable size block by simply applying one operation as often as needed.

In confusion layers, if the size of the block is set at design-time, we can simply have a separate confusion element for each position in the block (typically, each will be a distinct keyed substitution table).

However, in confusion layers where the size of the block is to vary dynamically during operation, we might select one from among many different processing elements. For example, we might select a substitution table to use at each processing position on the basis of position (in the block), some current data value (such as the value of the block at that position or the previous position), both of these, or some other quantity.

It is often convenient to think of each layer as an independent entity which updates the contents of the previous data block. But some forms of diffusion layer may connect to an earlier or subsequent layer. If realized only in layers of simple operations, such designs are likely to imply a (temporary) need for additional data storage. This can often be avoided by encapsulating the communicating operations in a single more-complex layer which may contain both confusion and diffusion.

Mechanism

The logical concept of a machine, which may be realized either as a physical machine, or as a sequence of logical commands executed by a physical machine.

A mechanism can be seen as a process or an implementation for performing that process (such as electronic hardware, computer software, hybrids, or the like).

One Way Diffusion

In the context of a block cipher, a one way diffusion layer will carry any changes in the data block in a direction from one side of the block to the other, but not in the opposite direction. This is the usual situation for fast, effective diffusion layer realizations.

In the context of a variable size block cipher, one way diffusion must conduct a change in one element to all succeeding elements, no matter how large the block.

This emphasis on the effects caused by a change in the input data occurs because even a single bit change to the input block should cause about half of the bits in the resulting ciphertext block to change state. See overall diffusion.

Overall Diffusion

That property of an ideal block cipher in which a change of even a single message bit will change every ciphertext bit with probability 0.5. In practice, a good block cipher will approach this ideal. This means that about half of the output bits should change for any possible change to the input block.

Overall diffusion means that the ciphertext will appear to change at random even between related message blocks, thus hiding message relationships which might be used to attack the cipher.

Overall diffusion can be measured statistically in a realized cipher and used to differentiate between better and worse designs. It does not, by itself, define a good cipher.

Permutation

The mathematical term for an arrangement, or re-arrangement, of symbols.

The second part of substitution-permutation block ciphers: First, substitution operations diffuse information across the width of each substitutions. Next, permutation operations act to re-arrange the bits of the substituted result; this ends a single round. In subsequent rounds, further substitutions occur, and re-arrangement occurs, until the block is thoroughly mixed and overall diffusion achieved.

Plaintext

Plaintext is the original, readable message. It is convenient to think of plaintext as being actual language characters, but may be any other symbols or values (such as arbitrary computer data) which need to be protected.

Polyalphabetic Combiner

A combining mechanism in which one input selects a substitution alphabet (or table), and another input selects a value from within the selected alphabet, said value becoming the combining result. Also called a table selection combiner.

Polyalphabetic combining is a poor cousin to Latin square combining, since a group of randomly-organized tables is unlikely to produce a balanced combiner. A Latin square will only include tables which achieve overall balance.

Random Number Generator

A random number generator is a standard computational tool which creates a sequence of apparently unrelated numbers which are often used in statistics and other computations.

In practice, most random number generators are computational mechanisms, and each number is directly determined from the previous state of the mechanism. Such a sequence is often called pseudo-random, to distinguish it from a sequence somehow composed of actually unrelated values.

A computational random number generator will always generate the same sequence if it is started in the same state. So if we initialize the state from a key, we can use the random number generator to shuffle a table into a particular order which we can reconstruct any time we have the same key.

Round

In the context of block cipher design, a term often associated with a Feistel block cipher such as DES. A round is the set of operations which are repeated multiple times to produce the final data. For example, DES uses 16 generally identical rounds, each of which performs a number of operations.

Secrecy

One of the objectives of cryptography: Keeping private information private.

Secret Code

A coding in which the correspondence between symbol and code value is kept secret.

Simple Substitution

Perhaps the original classical form of cipher, in which each plaintext character is enciphered as some different character. In essence, the order of the alphabet is scrambled, and the particular scrambled order (or the scrambling process which creates that particular order) is the cipher key. Normally we think of scrambling alphabetic letters, but any computer coding can be scrambled similarly.

Small, practical examples of Simple Substitution are easily realized in hardware or software. In software, we can have a table of values each of which can be indexed or selected by element number. In hardware, we can simply have addressable memory. Given an index value, we can select the element at the index location, and read or change the value of the selected element.

A substitution table will be initialized to contain exactly one occurrence of each possible symbol or character. This allows enciphering to be reversed and the ciphertext deciphered. For example, suppose we substitute a two-bit quantity, thus a value 0..3, in a particular table as follows:

```
2  3  1  0.
```

The above substitution table takes an input value to an output value by selecting a particular element. For example, an input of 0 selects 2 for output, and an input of 2 selects 1. If this is our enciphering, we can decipher with an inverse table. Since 0 is enciphered as 2, 2 must be deciphered as 0, and since 2 is enciphered as 1, 1 must be deciphered as 2, with the whole table as follows:

```
3  2  0  1.
```

Mathematically, a simple substitution is a mapping (from input to output) which is one-to-one and onto, and is therefore invertible.

Stream Cipher

In a conventional stream cipher, each element (for example, each byte) of the message is ciphered independently.

In a few stream cipher designs, the value of one message byte may change the enciphering of subsequent message bytes, but it cannot change the enciphering of previous message bytes.

In contrast, in a block cipher, changing even later bits in a block will generally change earlier bits of the enciphering. With a conventional 64-bit block this may be somewhat less interesting than finding the same property in a 1600-bit block.

Substitution Table

A linear array of values, indexed by position, which includes any value at most once. In cryptographic service, we normally use binary-power invertible tables with the same input and output range. For example, a byte-substitution table will have 256 elements, and will contain each of the values 0..255 exactly once. Any value 0..255 into that table will select some element for output which will also be in the range 0..255.

For the same range of input and output values, two invertible substitution tables differ only in the order or permutation of the values in the table.

A substitution table of practical size can only be thought of as a very weak cipher alone. But it can be a good part of a combination producing a stronger cipher.

Table Selection Combiner

A combining mechanism in which one input selects a table or substitution alphabet, and another input selects a value from within the selected table, said value becoming the combined result. Also called a polyalphabetic combiner.

Variable Size Block Cipher

A block cipher which supports ciphering in blocks of various size. The block size may vary only in steps of some element size (for example, a byte), but could be arbitrarily large.

Three characteristics distinguish a true variable size block cipher from designs which are imprecise about the size of block or element they support or the degree to which they support overall diffusion:

1. A variable size block cipher is indefinitely extensible and has no theoretical block size limitation;
2. A variable size block cipher can approach overall diffusion, such that each bit in the output block is a function of every bit in the input block; and
3. A true variable size block cipher does not require additional steps or layers to approach overall diffusion as the block size is expanded.

Variable Size Block Cipher Mechanism

A cryptographic mechanism which performs the role of a variable size block cipher.

Variable Size Confusing Means

A variable size block cipher layer which performs confusion. See confusion layer.

Variable Size Diffusing Means

A variable size block cipher layer which primarily performs diffusion across a block of arbitrary size, typically in one direction only. Such a layer must guarantee to conduct a change in one element to all succeeding elements, no matter how large the block.

For example, integer addition across the whole block does not do this, because a carry occurs to each next element only about half the time. That is, changes in less significant digits often do not affect more significant digits.

In contrast, an exclusive-OR chain is an example of a diffusion layer which will collect any input change, and conduct it to all succeeding elements. It is not surprising that any layer consisting of balanced cryptographic combiners, possibly intermixed with substitutions, can do this. The surprising thing is that this is a worthwhile thing to do.

Variable Size Layers

A block cipher layer which can be easily extended (in element-size steps) to arbitrary size.

Variable size layers must maintain specific properties during extension, so that a cipher using such layer need not use additional layers when larger blocks are ciphered.

Variable Size Substituting Means

A variable size block cipher layer which primarily performs confusion, typically by translating individual bytes of the block through one or more Simple Substitution tables. See confusion layer.

Variable size substituting means can support blocks of any design size by using a different substitution table for each processing position. But when the block size may vary dynamically, it is necessary to define which table will be used at each possible processing position. Perhaps the available tables would simply be used and re-used in sequence, or perhaps some mathematical function would select a particular table at pseudo-random from a group of tables.

Conclusion, Ramifications and Scope of Invention

Although the description of my invention contains many particulars, these should not be construed as limiting the scope of the invention. Specific examples variable size block ciphers are given merely to provide illustrations of the preferred embodiments of this invention. For example, the data paths and tables in the examples can be some size other than byte-size, and the block size can be arbitrarily large and even dynamic. Layers can be added as needed, either like those already in a particular example, or from another example, or other layers using any other sort of variable size layer technology. One or more layers can be added to perform dynamic keying without expanding the block size. Layers can also be eliminated to improve speed, while perhaps lowering the quality of the cipher.

Even though the principle cryptographic use for these particular mechanisms is as ciphers, they can also be used to create one-way functions and hash operations. Consider an authentication example: First we start with a particular block value. Simply by feeding part of the block back to the input, along with some amount of new text, a new block or authentication value can be produced by ciphering. By repeatedly processing text in this way, an authentication value can be developed covering all the text so processed. Because the mechanism can be made as wide as desired, the authentication value can be as large as desired.

The scope of the invention should be determined by the appended claims and their legal equivalents, rather than by the specific examples given.

Claims

1. A cryptographic mechanism for block ciphering a data block of any size, said mechanism consisting of a fixed number of layers, including two one way diffusion layers.

2. The cryptographic mechanism of claim 1 wherein one of said one way diffusion layers includes confusion operations.

3. The cryptographic mechanism of claim 1 wherein two of said one way diffusion layers diffuse information in opposite directions.

4. The cryptographic mechanism of claim 1 wherein one of said one way diffusion layers uses carry information produced by an earlier one way diffusion layer.

5. The cryptographic mechanism of claim 1 wherein two of said one way diffusion layers include combining mechanisms selected from the group consisting of exclusive-or, addition, subtraction and additive combining.

6. The cryptographic mechanism of claim 1 wherein one of said one way diffusion layers includes combining mechanisms selected from the group consisting of table selection, polyalphabetic combining, latin square combining, balanced block mixing and dynamic substitution.

7. A variable size confusion layer cryptographic mechanism, including, in combination:

(a) a fixed number of substitution means for processing block elements, and
(b) selection means for selecting a particular substitution means from among said fixed number of substitution means,

wherein at each position in said variable size layer a substitution means is selected, whereby a data block of any number of elements may be confused.

Terry Ritter, his current address, and his top page.

Last updated: 1998-03-10

A Variable Size Core for Block Cipher Cryptography

Cipher Blocks of Arbitrary Size with Good Data Diffusion

A *Ciphers By Ritter* Page

Terry Ritter

A Variable Size Block Cipher (VSBC) is distinguished by its ability to cipher blocks of dynamically arbitrary size, to the byte, and yet produce good data diffusion across the whole block.

In dramatic contrast to conventional designs, here most diffusion occurs in simple *linear* BBM operations **which are not intended to provide strength!** By itself, linear mixing has no strength at all, but it does set the stage for keyed substitution tables which *do* have strength. The BBM's provide a *guaranteed,* exactly balanced and reversible mixing which, in a successful design, prevents individual tables from being distinguished and solved.

Here we present the ciphering ***core,*** the architecture of tables and mixing which performs actual ciphering. The tables are initialized or "keyed" by a separate process which shuffles each table independently, twice.

Variable Size Block Cipher Attributes

- **SPEED** -- about 985,000 bytes per second in 16-byte blocks and about 1,120,000 bytes per second in 64-byte blocks (on a 100 MHz "686," under Win95, with the Borland 32-bit Delphi 2 compiler). The typical keying overhead is around 64 milliseconds for 64 tables.
- **STRENGTH** -- greater than 128 bits for a block size of at least 64 bits and at least 64 keyed tables.
- **BLOCK SIZE** -- dynamically selectable to the byte at ciphering time, but this is a block cipher and ***not*** a stream cipher. Clearly, a single program can handle "legacy" 64-bit blocks, "modern" 128-bit blocks, "independent" 64-byte blocks and 512-byte disk blocks, and anything in between.
- **KEY SIZE** -- arbitrary, independent of block size.

Contents

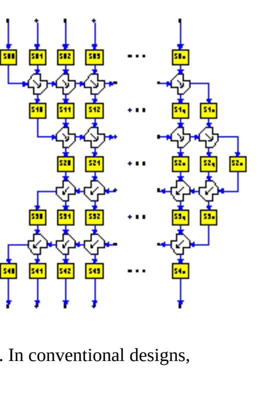
- Variable Size Block Cipher Attributes
- Background
- Components
- Scalability
- The BBM
- A Variable Size Block
- Dynamic Table Selection
- Data Diffusion
- Deciphering
- Large Blocks
- Primary Keying
- Strength
- Dynamic Keying
- Authentication
- Strength Arguments
- Summary of Advantages
- Also See:

Background

The first known publication of the concept of a Variable Size Block Cipher (VSBC) -- and the first published VSBC design -- was the 1995 Aug 20 sci.crypt announcement by Ritter and the resulting discussion. The general lack of understanding in the responses indicates that the concept was new to those who replied.

A patent has been granted on Variable Size Block Ciphers.

Components



In this Variable Size Block Cipher core we have two main component types:

1. keyed, invertible substitution tables, and
2. Balanced Block Mixers (BBM's).

(We also use exclusive-OR's in the dynamic table selection.)

In the figure, each box represents a table, and each hourglass shape represents a BBM. Plaintext data bytes enter at the top, are substituted, mixed right, substituted, mixed right, substituted, mixed left and substituted into ciphertext. Dynamic table selection is not shown.

The essence of the design is an inability to externally isolate or distinguish one table from another. This inability is directly related to the exact balance in the high-quality BBM mixing, as opposed to an exponential increase in complexity from repeated low-quality mixing with confusion.

Tables are widely used in many designs, and even keyed tables are not that unusual. But in this design *all* of the inter-element diffusion occurs in distinct, one-way diffusion layers composed of *linear* Balanced Block Mixing (BBM) operations, and this *is* new.

This separation between mixing (diffusion) and keyed substitution (confusion) is beneficial in that the two operations can be seen, understood, and measured separately. In conventional designs, confusion and diffusion are often combined, which makes these ciphers difficult to understand *or* measure. And conventional ciphers do not scale down to testable size.

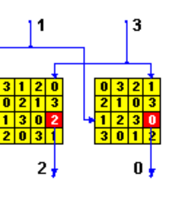
Scalability

A Variable Size Block Cipher is scalable in a way that few designs are:

- First, the block size is dynamically selectable to the byte. A single unchanged program can handle "legacy" 64-bit blocks, "modern" 128-bit blocks, "independent" 64-byte (512-bit) blocks, *and* everything in between.
- Next, we can even scale down the size of the tables and the mixing elements. This means that the same design can deliver either a full-scale real cipher, or a tiny model which can be investigated experimentally for weakness.

This ability to scale a single design between large real ciphers and small testable models almost cannot be overemphasized. Cryptography **does** have better-known structures and designs, but cryptography simply **does not** have a proven methodology which -- if only followed -- will guarantee a strong design. This means that testing is exceedingly important, and real ciphers are far too large for many types of test.

The BBM

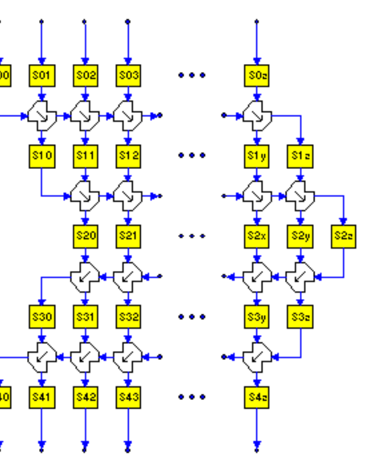


A Balanced Block Mixer can be considered an orthogonal pair of Latin squares. The two inputs select "row" and "column" and return the value from the selected element of each square. This "mixes" or combines the two input values into the two output values with some useful properties for this design:

1. A change to either input will change both outputs.
2. No input change can keep both outputs unchanged.
3. Any possible value (value balance) from either output can be achieved by changing either input (structural balance), regardless of the value on the other input (independence).

Even a single bit change in one mixing input will produce some change in both mixing outputs. And if each output is substituted through a keyed table, each table will produce a random result value. So even a single bit change on one input will produce a new random result on both outputs.

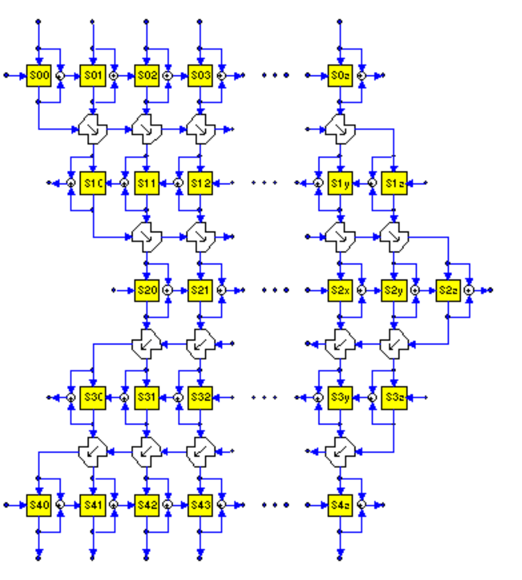
A Variable Size Block



To have a block cipher which can be expanded arbitrarily, to the byte, at ciphering time, we are necessarily limited to operations which can be applied byte-by-byte. But we are *not* limited to performing the whole transformation on each byte *separately* as we would in a stream cipher. Instead, we can apply operations which, though byte-size, produce "carry" information as an input to the *next* byte operation. When we do this multiple times in both directions we can approach the desired block cipher goal of apparent overall diffusion.

In the figure, each plaintext byte enters the top of the structure and is immediately confused through an invertible substitution table. The substituted value then mixes into a one way diffusion path running across the unknown width of the block. Each of the funny-looking hourglass shapes represents a separate Balanced Block Mixing (BBM), a two-input two-output mixing operation which proceeds in the direction of the arrow.

Dynamic Table Selection

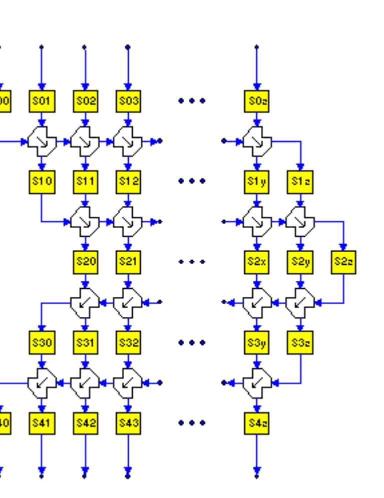


While each substitution in the figures has a particular number (e.g., S00 for the leftmost top substitution), these are just *positions* for tables and not particular *tables* themselves. Considerable strength is added by selecting the table for each next position based on an exclusive-OR of the data and substituted data values in the preceding position. This "dynamic table selection" virtually eliminates the possibility of having even two blocks ciphered under the same table arrangement.

Dynamic table selection also provides a zero-latency form of "chaining" which can be used for authentication. (The initial table values can be explicitly reset before each block if blocks are to be ciphered independently.)

Data Diffusion

The intent of the design is to mix every input byte into every output byte in a way which appears to be a huge overall random substitution. Most of this mixing occurs in the four BBM one-way diffusion layers, and some occurs in the dynamic table selection of the confusion layers.



While one might think that two one-way diffusion channels would be sufficient (one each way, or both the same way with an end-around-carry), in practice, random inputs occasionally "cancel" earlier changes, so that -- occasionally -- no propagation occurs. This can be made rare by having a wide diffusion channel. And here we have not only two layers of Balanced Block Mixing in each direction, but also the dynamic table selection to carry diffusion across the block. In this particular design, there are 4 specific 8-bit one-way diffusion layers, plus 5 layers of dynamic table selection. With 64 substitution boxes, we thus have a 62-bit diffusion path between adjacent columns.

The first two right-going BBM layers assure that almost any left-side data change will be conducted across the block, and so complicate attempts to isolate the leftmost column. After the first left-going diffusion layer, we should have a complex function of *every* input byte across the entire block. The final left-going BBM layer is used to protect against random cancellation effects which might otherwise affect the left-going diffusion.

By conducting a defined plaintext attack, and running through all values for the first few columns, it should be possible to detect cancellations in both carry paths as a lack of ciphertext changes to the right. These are all evenly distributed, however, and seem to provide no information about particular values which could be used in reconstructing the table contents. This issue is discussed in Defined Plaintext Attack on a Simplified BBM VSBC on these pages.

Deciphering

Since we use invertible substitution tables, we can "undo" any table translation by sending the resulting value through the corresponding inverse table. The form of Dynamic Table Selection used here takes both the input and output table values and exclusive-OR's them, which selects the same next table whether we have forward or inverse tables. So a right-going Dynamic Table Selection layer is reversed by a right-going Dynamic Table Selection layer using inverse tables.

The inverse of the simple linear BBM component used here is just another occurrence of that same BBM component. In the one-way diffusion layers used here, a left-going diffusion (L) is the inverse of a right-going diffusion (R), and vise versa. In the current design we have four one-way diffusion layers for enciphering, in the sequence <R R L L>. When deciphering, to reverse the bottom "L" layer, we need an "R", and so on. So the deciphering diffusion layer sequence is *also* <R R L L>.

This means that *exactly the same ciphering routine* can be used for both enciphering and deciphering. For deciphering we use inverse tables and take the initial values (IV's) in reverse layer order. Both of these conditions are easily handled during keying.

Large Blocks

DES has a fixed block size and we have somehow managed thus far, so it may seem like overkill for a cipher to have multiple block sizes. But there is almost no cost for this, and it must be admitted that having blocks of various size -- or even dynamically variable size -- sometimes can provide a better fit to the overall system than any single fixed block size. But perhaps the biggest benefit comes from the ability to cipher in very large blocks.

Large Blocks are Strong

If plaintext really does contain uniqueness at a rate of only about one bit per character, a legacy 64-bit block covers only about eight bits of uniqueness. This is the situation encountered in the classic codebook attack. This sort of attack is *not* avoided by having a larger keyspace, but *can* be avoided by using a wide, unbiased selection of plaintext blocks. Normally this is done by using a chained operating mode to randomize the plaintext. But increasing the block size to 64 bytes or more can collect enough uniqueness in the plaintext block so that randomization can be avoided.

Large Blocks Can Be Independent

By increasing the block size to 64 bytes or more we may be able to operate in "electronic code book" (ECB) mode instead of "cipher block chain" (CBC) mode. This means that we may not need to develop, send or store an initial value (IV), which would otherwise expand the ciphertext. And it also means that blocks can be both enciphered and deciphered independently, even if they are received out-of-sequence, as may happen in packet-switching transmission.

Large Blocks Have Room for Dynamic Keying and Authentication

In conventional block cipher designs, the block size is so small that we can scarcely consider displacing some data with other information. But when we have a large block, there is room for other information, at a relatively small overhead. Typical applications include Dynamic Keying and Authentication.

Primary Keying

Primary keying generally consists of shuffling each substitution table with a keyed cryptographic random number generator (RNG), twice. Primary keying can use fairly conventional technology, and is largely independent of the ciphering core itself. One example of a keying system is described in A Keyed Shuffling System on these pages.

Primary keying takes about 1 msec per table on a 100 MHz processor, which is fast enough on a human scale. Primary keying does take time, because it is specifically intended to perform as much computation as possible -- once -- instead of having computation repeated with every block at ciphering-time.

Strength

We assign an 8 bit strength to each table. Although an arbitrary table permutation contains 1684 bits of independence, we reason that if The Opponents can find (that is, absolutely identify and confirm) one value, they can probably find another. Since each table value is 8 bits, we assume a table strength of just 8 bits.

With this sort of five confusion layer and four one-way diffusion layer structure, we believe that only three of the layers contribute independent strength. Therefore, for a small 8-byte block, we expect to see 8 * 24 = 192 bits of strength in 40 separate tables (out of at least 64).

Dynamic Keying

True zero-latency dynamic keying is available by placing keying values in each data block along with data. This will of course expand the ciphertext by the size of the keying field, but even a 64-bit dynamic keying field is only about 12.5 percent of a 64-byte block. This sort of keying can be used in any true (that is, avalanching or data diffusing) block cipher with room for extra data.

Authentication

Strong block-by-block authentication is available similar to dynamic keying. Authentication values are placed into each data block along with data. Potentially, this can avoid a higher-level scan across the data with a cryptographic hash function. The exact same field can provide both authentication *and* dynamic keying.

Strength Arguments

Here we present various attacks and comment on their likelihood of success on this particular cipher. Recall that attacks are not *algorithms,* but instead just general approaches which must be reinvented for every new type of cipher.

Exhaustive Search (Brute Force on the Key)

> Try each possible key until the message deciphers properly. Try most-likely keys first.

A keyspace of at least 120 bits should be sufficient to prevent exhaustive search in the foreseeable future. The keying system for the VSBC core has a keyspace substantially beyond this value, mainly because this produces a convenient design.

No cipher can do very much about key search attacks if there are only a relatively small number of possible keys, and if some keys are vastly more probable than others. It is the responsibility of the larger system to prevent this.

Chosen Key

> Try various keys on known plaintext and compare the resulting ciphertext to the actual ciphertext, to try and build the correct key value.

If a user has the ability to generate specific keys which are used by the VSBC core on data, it is quite likely that the external cipher system has already failed. However, even in this situation, key selection seems unlikely to help The Opponent. Sure, they can force particular table values by manipulating the key, but they can do that *without* going through the keying process. The Opponent's main problem in attacking the cipher is that the one-way diffusion layers appear to couple the various tables together so that a single table cannot be isolated and worked on separately.

Ciphertext Only

> The Opponent accumulates a mass of ciphertext material and tries to find relationships within the data.

This is a general class of various specialized attacks which all use only the exposed ciphertext as opposed to particular knowledge of the plaintext or access to the ciphering system itself.

Ciphertext-Only Codebook

> Collect as many ciphertexts as possible and try to understand their contents through usage and relationships; then, when a ciphertext occurs, look it up. This treats the block cipher like a *code,* and is the classic approach to code-breaking.

Just as some letters are more frequently used than others, words and phrases also have usage frequencies, as do blocks which contain plaintext. If the cipher block size is small (under 64 bytes), and if the plaintext is not randomized, and if dynamic keying is not used, and if the ciphering key is not changed frequently, it may be possible to build a codebook of block values with their intended meanings.

Codebook attacks of any sort are ideally prevented by having a large number of block values, which implies a large block size. Once the block size is at least, say, 64 bytes, we expect the amount of uniqueness in each block to exceed anyone's ability to collect and form a codebook.

Since the complexity of any sort of a codebook attack is related to block size only, doing "triple" anything will not affect increase this complexity. In particular, this means that Triple DES is no stronger that DES itself under this sort of attack, which is based on block size and not transformation complexity.

Known Plaintext

> Somehow "obtain" both the plaintext and the corresponding ciphertext for some large number of encipherings under one key.

First, since the VSBC core described here has an internal state typically 512 times as large as a 64-byte data block, we know that a single plaintext and ciphertext pair simply do not contain sufficient information to reveal the full internal state. Note that a *single* known plaintext and ciphertext pair probably *would* identify a DES key.

Larger amounts of known plaintext and ciphertext will of course surpass the required information, but the question is how The Opponent might use it. The problem is a supposed inability to distinguish one table from another and so work on one table at at time.

Known-Plaintext Codebook

> Collect as many ciphertexts and associated plaintext blocks as possible; then, when a ciphertext occurs, look it up.

Small block ciphers prevent codebook attacks by randomizing the plaintext (often with Cipher Block Chaining) so that the plaintext block values are distributed evenly across all possible block values. But not all block ciphers are always applied properly.

Codebook attacks are ideally prevented by having a large number of block values, which implies a block size. To prevent this attack for the future, we need a block size of at least 128 bits, and even then *still* require the plaintext to be randomized. If we wish to avoid randomizing with CBC, we need a block which is large enough so the uniqueness it does contain assures that there will be too many different blocks to catalog. A reasonable power-of-2 minimum size to avoid randomization would be at least 64 bytes.

Chosen Plaintext

> Without knowing the key, arrange to cipher data at will and capture the associated ciphertext. Dynamically modify the data to reveal the key, or keyed values in the cipher.

The point here is not to decipher the associated ciphertext because The Opponent is *producing* the original plaintext. If The Opponents have chosen plaintext capabilities, they can probably also submit arbitrary ciphertext blocks for deciphering.

The weakness to be exploited here depends upon the ciphering system beyond the core cipher *per se.* If the primary keying values change with each message, and the ciphering keys are not under the control of the user (if the system uses message keys), there simply *is* no fixed internal state to be exposed.

If the primary key remains the same for all messages, then there will be some keyed state to try and ferret out. But if the ciphering system uses dynamic keying fields (with values again not under the control of the user), there can be no completely-known Chosen Plaintext blocks for use in analysis.

If the ciphering core is used raw, without primary re-keying and also without dynamic keying, the Dynamic Table Selection still uses whatever "uniqueness" exists in the data to change the tables used in the block transformation.

It is not clear that there exist any statistical relationships in the VSBC core which can be exploited better by Chosen Plaintext than by Known Plaintext.

Chosen-Plaintext Codebook

> Create as many ciphertexts and associated plaintext blocks as possible; then, when a ciphertext occurs, look it up.

This is much like the previous codebook attacks, now with the ability to fill the codebook at will and at electronic speeds. Again, the ability to do this depends upon the cipher having a relatively small block size and not having a dynamic keying field. How difficult such an attack would be depends upon the size of the block, but a VSBC may *have* no particular block size.

Meet-in-the-Middle

> With a multi-layered structure, given known-or defined-plaintext, search the top keyspace to find every possible result, and search the bottom keyspace to find every possible value.

With a two-level construct and a small block size, matches can be verified with a few subsequent known-plaintext/ciphertext pairs. Of course, three and more-level constructs can always be partitioned into two sections so a meet-in-the-middle attack can always be applied; this just may be pretty complex.

The Mixing core avoids meet-in-the-middle attacks by using a three-level construction, in which each layer has a huge amount of keyed state or "keyspace."

Key Bit Bias

> Through extensive ciphering of fixed plaintext data under a variety of different keys, it may sometimes be possible to associate key bits with the statistical value of some ciphertext bits. This knowledge will break the cipher quickly.

This is a rather unlikely circumstance, albeit one with absolutely deadly results.

Differential Cryptanalysis

> Exploit known properties of particular known substitution tables to effectively reduce the number of "rounds" in an iterated block cipher.

The original form of Differential Cryptanalysis mainly applies to iterated block ciphers with known tables, neither of which are present here. However, the general concept is like the expected VSBC attack.

The general idea of trying to relate byte values in adjacent columns is one of the major possible attacks on VSBC's. This is made difficult by using Balanced Block Mixing to conduct diffusion, and by having multiple diffusion layers. Each BBM guarantees that a change in a data value can "cancel" at most *one* of the mixed outputs, which leaves the other mixed output to carry the change across the block.

Another difficulty in applying Differential Cryptanalysis to a VSBC would be in handling the Dynamic Table Selection feature. Each different input value will actually select a different set of tables, and, thus, a completely different transformation. It is hard to attack a transformation which changes its structure whenever it is probed.

Summary of Advantages

Strength

- The strength advantage of large blocks, which can hold huge amounts of plaintext uncertainty.
- Large blocks have room for dynamic keying information, which supports zero-latency block-by-block keying.
- Large blocks have room for authentication information, which can avoid the need for a slow higher-level authentication pass over the data
- The strength advantage of massive, keyed, nonlinear, hidden internal state, which generally means that an attacker must somehow *expose* that state.
- The strength advantage of dynamic table selection, since there exists no fixed cipher to be attacked, even for an unchanging key.
- The strength advantage of dynamically selectable block size to the byte, which supports the random block-partitioning of data (as opposed to the fixed-size blocks in a standard block cipher).
- The strength advantage of a fundamentally *scalable* design which supports tiny true models that can be exhaustively tested.

Flexibility

- The flexibility advantage of dynamically selectable block size. This directly supports "legacy" 64-bit blocks, "modern" 128-bit blocks, "independent" 64-byte blocks, and 512-byte disk sectors in *the exact same unchanged program.*
- Large blocks can eliminate the need for plaintext randomization, chaining, and the use of ciphertext-expanding IV's.
- Large blocks support the independent ciphering of blocks, and random-access ciphering.
- The dynamically selectable block size can cipher *existing* data structures in *existing* systems *without* needing extra data storage and resulting modification of the system.
- The dynamically selectable block size to the byte need not expand data *at all.*
- The flexibility advantage of key processing as a part of the cipher. This directly supports *both* textual *and* random binary keys of arbitrary length.
- The flexibility advantage of *the ideal single cipher,* since it efficiently covers ciphering jobs of all block sizes.

Also see:

- The Large Block DES Open Development: our search for a new "DES" architecture started over four years ago.
- The original BBM article
- The start of a theoretical basis
- Realized VSBC prototypes
- The current BBM article
- The keying system
- The one-way diffusion layers
- VSBC's on the top page

Terry Ritter, his current address, and his top page.

Last updated: 1998-03-21

Efficient One-Way Mixing Diffusions

Diffusion Across Large Blocks Using Byte-Size BBM's

Terry Ritter

The use of Balanced Block Mixers is new in cryptography, as are one-way diffusion layers. Here we present description, figures, and routines in both Pascal and C for variable-size one-way diffusion layers using Balanced Block Mixing technology. These handle an arbitrary block size as selected at ciphering time.

Source code is presented for bona fide informational or experimental purposes only: The mechanism described is protected by patent.

Mixing in Block Ciphers

Good data mixing is at the heart of block cipher design. Normally, we expect every input bit to be mixed into every output bit in some complex way. In Mixing ciphers based on Balanced Block Mixing (BBM) operations in FFT patterns, this can be achieved in log n sublevels of n operations each, but only for power-of-2 size blocks. In a Variable Size Block Cipher (VSBC), this is not good enough, because we want to handle blocks of dynamically *arbitrary* size, to the byte.

The original VSBC proposal chained data values across a block with exclusive-OR. This handled diffusion, and arbitrary block size, but may be easier to attack. Since the exclusive-OR operation produces just a single result, it is easier to externally manipulate by introducing two adjacent data values which "cancel" the diffusion. This immediately leads to establishing relationships between tables. Dynamic table selection may complicate this, but it is important to provide independent strength on multiple levels.

The BBM component seems better at resisting external manipulation, because if one of the data values is changed, there is no second data value which can cancel *both* outputs from the BBM. If we could assure that *both* outputs are sent across the block unchanged, we could guarantee that any input change whatsoever would produce overall diffusion. But because of the limited width of each diffusion channel, it is possible for diffusion to be canceled by chance. This is the reason for having multiple diffusion channels, and multiple diffusion mechanisms.

The Mixing Component

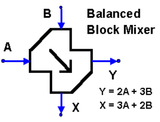
Here we have the schematic representation for the BBM operation described in other articles (e.g., the current BBM article). Basically, we have a two-input two-output operation which is invertible.

In particular we have the equations shown in the figure, computed in mod-2 polynomials with some appropriate irreducible p. We say the results are computed "mod-2, mod p".

There are two immediate consequences of this construction:

- Changing *one* of the inputs is guaranteed to change *both* of the outputs.
- By changing *both* inputs, we can force either of the outputs to remain unchanged, but then *the other* output *will* change.

These consequences help us reason about the flow of diffusion and the ability to attack the result.

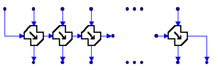


Mixing To The Right

Here we have a typical BBM one-way diffusion layer. We first note that the input block size (the number of data paths on the top) is the same as the output block size (the data paths on the bottom). Further, no Initial Values are required.

Basically, this particular layer collects the uniqueness present in the input, and transports across the block. Clearly, the amount of uniqueness transported is limited by the size of the diffusion path, here a single byte. But any uniqueness to the left will be carried across the full block if no other uniqueness intrudes. And if other uniqueness *does* intrude, we have transported the information up to that point, and the fact that we eventually exceed the information limits of the diffusion path may not matter.

Clearly, the amount of transported diffusion is limited by the size of the diffusion path, which is ample reason to have multiple diffusion layers.

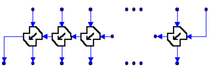


Right-Going BBM One-Way Diffusion Layer

Mixing To The Left

Here we have the left-going analog to the right-going layer described above. This particular left-going diffusion layer is also the inverse of the earlier right-going diffusion layer. By feeding the output of the right-going layer into the left-going layer, we can "undo" what the first layer did. This of course gives us a way to build decipherring operations.

A full VSBC design is typically composed of multiple one-way diffusion layers, plus confusion layers containing keyed "byte-wide" substitution tables. The designer first has the task of seeing that any uniqueness is transported to one side of the block. Then the task is to move it back the other way, all the way to the *other* end of the block. (Or, in other designs, perhaps around again via an intermediate carry).



Left-Going BBM One-Way Diffusion Layer

Pascal Mixing Routines

Here we show routines in Pascal for providing one-way diffusion across blocks of dynamically arbitrary size. Dynamic block size selection supports, for example, the general use of large (say, 64 byte) blocks until the end of the message, when perhaps two smaller blocks will be produced, without expanding the ciphertext.

Note that the simple Balanced Block Mixing computations are part of the routine. Here we use the degree-8 polynomial 01a9h.

```
PROCEDURE BBMixRt( VAR buf; byct: WORD );
   { buf[0] -> c; BBM(c,[i]) -> [i-1],c; c -> [byct-1] }
   { undo BBMixRt with BBMixLt }
   VAR
      ba: ByteArray ABSOLUTE buf;
      i, t: WORD;
      a, b: BYTE;
   BEGIN
   IF (byct < 2) THEN  Exit;

   a := ba[0];

   FOR i := 1 TO PRED(byct) DO
      BEGIN
      b := ba[i];
      t := (a XOR b) Shl 1;          { 2a + 2b }
      IF ((t AND $100) <> 0) THEN
         t := t XOR $1a9;            { 2a + 2b - p }
      ba[PRED(i)] := t XOR a;        { 3a + 2b }
      a := t XOR b;                  { 2a + 3b }
      END;

   ba[PRED(byct)] := a;
   END; {BBMixRt}



PROCEDURE BBMixLt( VAR buf; byct: WORD );
   { buf[byct-1] -> c; BBM([i],c) -> c,[i+1]; c -> [0] }
   { undo BBMixLt with BBMixRt }
   VAR
      ba: ByteArray ABSOLUTE buf;
      i, t: WORD;
      a, b: BYTE;
   BEGIN
   IF (byct < 2) THEN  Exit;

   b := ba[PRED(byct)];

   FOR i := byct-2 DOWNTO 0 DO
      BEGIN
      a := ba[i];
      t := (a XOR b) Shl 1;          { 2a + 2b }
      IF ((t AND $100) <> 0) THEN
         t := t XOR $1a9;            { 2a + 2b - p }
      ba[SUCC(i)] := t XOR b;        { 2a + 3b }
      b := t XOR a;                  { 3a + 2b }
      END;

   ba[0] := b;
   END; {BBMixLt}
```

C Mixing Routines

Here we show routines in C for providing one-way diffusion across blocks of dynamically arbitrary size.

Note that the simple Balanced Block Mixing computations are included. Here we use the degree-8 polynomial 01a9h. The BYTE and WORD defines are as one might expect.

```
/*
 * buf[0] -> c; BBM(c,[i]) -> [i-1],c; c -> [byct-1]
 *
 */
void
BBMixRt( BYTE buf[], WORD byct )
{
   WORD  i;
   BYTE  *bp = buf, *sp = buf, a, b, t;

   a = *bp++;

   for (i = 1; i < byct; i++)
      {

      b = *bp++;
      t = a ^ b;

      if (t & 0x80)
         t = (t + t) ^ 0xa9;
      else
         t = t + t;

      *sp++ = t ^ a;
      a = t ^ b;
      }

   *sp = a;
}



/*
 * buf[0] -> c; BBM(c,[i]) -> [i-1],c; c -> [byct-1]
 *
 */
void
BBMixLt( BYTE buf[], WORD byct )
{
   WORD  i;
   BYTE  *bp, *sp, a, b, t;

   bp = sp = buf + byct -1;

   b = *bp--;

   for (i = 1; i < byct; i++)
      {

      a = *bp--;
      t = a ^ b;

      if (t & 0x80)
         t = (t + t) ^ 0xa9;
      else
         t = t + t;

      *sp-- = t ^ b;
      b = t ^ a;
      }

   *sp = b;
}
```

Summary

We have developed a description and presented working routines in both Pascal and C for a particular efficient one-way diffusion.

There may be slightly more efficient ways to code this, although, as it stands, it does drop into 80x86 assembly language fairly well. Unfortunately, byte-level processing is not a big advantage of 32-bit code.

***Terry Ritter**, his current address, and his top page.*

***Last updated:* 1997-06-10**

Defined Plaintext Attack on a Simplified BBM VSBC

Tantalizing Opportunities, but Few Results

Terry Ritter

Introduction

Here we consider a ***Defined* Plaintext attack** on the structure of a Variable Size Block Cipher (VSBC) which uses Balanced Block Mixing (BBM) components. In this attack, The Opponent can introduce plaintext at will and collect the related ciphertext, but does not know the key or the structure of any table.

It is important to understand that the larger cipher system ***can*** and ***should*** make Defined Plaintext attacks **impossible** in many applications. This can be done by using random keying specific to each message:

- by using a different starting IV (table-select values) for each message,
- by using a different primary key for each message (classical message keys), or
- by placing different dynamic keying information in each block (modern message keys).

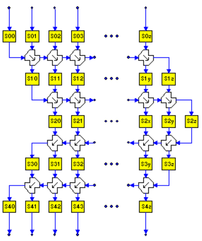
All of these approaches use random values which The Opponent can neither define nor expose, but these values must be available on both ends. One possibility is that both ends retain some secret but progressing cryptographic state which is used for message keys. But more typically, using message keys implies that random keying information (enciphered under a main key) must be transported (or saved) along with the message. The problem is that this *expands* the ciphertext. Of course, in many or even *most* applications, that is no problem at all.

But applications do exist for fast ciphering which does *not* expand ciphertext, and this may eliminate the above possibilities. A typical application might be the ciphering of variable-size data fields in a database program. This is exactly this sort of application which could most benefit from a VSBC, and yet also best supports a Defined Plaintext attack. We wish to see how successful such an attack might be.

The Real System

One reasonable approach to a VSBC design is described in A Variable Size Block Core on these pages. Not shown is the Dynamic Table Selection, where we select each particular table from an array of tables, based on both the data and table in the previous column. The diagram is already visually complicated by the many tables used, and here we wish to simplify that architecture and so better understand the ability to attack it.

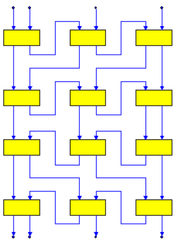
Perhaps the best way to simplify the diagram is to eliminate the tables, and to show a small fixed-size structure for analysis. We ignore Dynamic Table Selection to see what we have without it.



The Simplified System

Here we have the simplified architecture. We hide the tables without loss of generality simply by assuming that each BBM may have any desired structure. Each of these BBM's might be a separate pair of keyed, 65536-byte Latin square tables, for extreme strength and nonlinearity. But in practice, there is a limit on the effort we want to put into main keying, and this probably means using linear BBM's with smaller nonlinear tables. Unfortunately, attacking the system eventually means resolving those tables, and then the diagram is no longer simple anymore.

Still, we can illustrate some issues with a simplified diagram. Here we have 12 BBM's in 4 rows by 3 columns forming a 4-byte block cipher. The top two rows represent right-going one-way diffusion layers. In the simplified system (*without* Dynamic Table Selection), these are the *only* right-going diffusions, so if these can be defeated, the block-wide diffusion we expect from a block cipher can also be defeated.



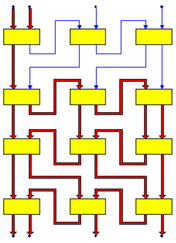
Cancelling Diffusion in One Layer

Suppose we try to "cancel" the right-going diffusion in the top layer. That is, suppose we *change* the leftmost data byte and then *also* change the adjacent data byte. In the figure, each "change," and each "consequence" of those changes, are highlighted as medium red lines.

Probably the best way to think about these structures is to recall some useful BBM properties:

- The mapping is one-to-one: Every possible input value (across both inputs) produces a different output value (taken across both outputs), and all values are possible.
- Any possible change to a single input will always change both outputs; to change only a single output, it is necessary to change both inputs.
- Stepping any one input through all possible values (while keeping the other input fixed) will step every output through all possible values.

Because of the BBM properties, we can always find some second byte value which will leave the top right-going diffusion channel unchanged, despite the data having changed. But there are *two* right-going diffusion layers, and the second will pick up the change and carry it across the block. The obvious next question is whether one can cancel *both* right-going layers simultaneously.



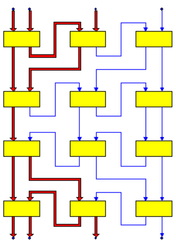
Cancelling Diffusion in Two Layers Simultaneously

We have seen that we *can* cancel the first right-going carry, so now we are interested in cancelling the carry from the the leftmost BBM in the second layer. For any change in the left input of that second-layer BBM, we can cancel that change (leave the carry-channel unchanged), *provided* that we can place any possible value on the right input. We can do *that* if we can control both of the inputs to the BBM in the second column in the top row. We do have direct control of the right input to the second BBM, but the left input is the carry from the first BBM in that row.

So the question becomes: If we traverse all possible values of the second and third bytes, can we find a pair which will leave both carry channels unchanged? This is the condition shown in the diagram. And this brings us back to the properties of the BBM.

In the same sense that each BBM is itself invertible, the chained structure with 3 inputs and 3 outputs is *also* invertible. This means that one can fix the values on any two output channels, and still support every possible value on the other. For *any* two carry values, there are *always* 2^8 triples (out of 2^{24} for the 3 inputs) which will leave the carry channels unchanged, and this is independent of whether the BBM's are linear or not. Presumably, one could externally detect and collect such "groups" of related values, and this can be done with *any* adjacent three columns, not just the start of the block.

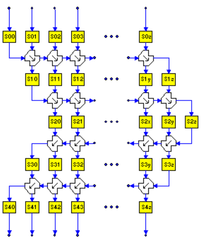
So, if we change the leftmost byte, there is exactly one pair of bytes which will produce the same two right-going carry values as before. And we can detect that situation because of a lack of avalanche in the right part of the block.



Attacking the Bottom-Left BBM's

Of all 2^{24} possible values of the left three bytes, there are 2^{16} "groups" of 256 triples which each represent a particular pair of right-going carry values. Within such a group, S20 can be taken through all possible values (although the actual values will not be known). Further, the left-going carry inputs to both the leftmost BBM on the third row, and the second BBM on the fourth row will be fixed, because all right-going diffusion has been cancelled.

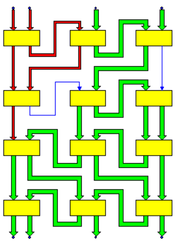
One can think of solving the bottom-left sub-system, or at least getting some relationships, because of the linear nature of the BBM's proper. But as far as is known, the keyed tables S30, S31, S40, S41, and S42 prevent those computations.



Extending the Relation

Assuming that we have a input triple that cancels all right-going diffusion (again, in the absence of Dynamic Table Selection) we want to know is whether we can *extend* the relationship for another input value, to produce a related quad.

In the figure, the mid-size red lines are the changed bytes with a successful cancelling set. The larger green lines represent our attempt to build upon that situation. To continue to cancel the right-going diffusion, we have to keep the right outputs fixed for two changing BBM's in the top two layers. But this is harder than it looks. If we try to extend the situation by only changing the third and fourth bytes, there seems to be just no way to do it.



Comments

The ability to cancel the right-going diffusions seems to set out a substantial relationship in the cipher, but it is unclear how this can be used. The internal linear relationships are hidden behind layers of keyed tables, so the actual values involved are not known.

In analysis we seek situations in which some values do not occur, or in which some values occur at least twice, for we can use that information to define a value or relationship. But when every value can always occur with the same probability, that particular sort of reasoning does not seem helpful.

In fact, this ability to cancel the right-going diffusions seems little more than an obvious extension of the property of a single BBM: Suppose we have just one BBM, with a table on each input and a table on each output. Even in this minimal case, we can obviously select input values which change only the leftmost output, so that no "avalanche" occurs in the rightmost element. Presumably this gives us information about the system, but does this really help us expose any table element, or even yield any useful relationship at all? And if not, can we not reason that the more complex multi-layer structure must in fact behave similarly?

Summary

First we again note that we discuss a *simplified* system in which Dynamic Table Selection is not part of the structure. The three layers of right-going Dynamic Table Selection in a practical design should complicate this sort of attack by up to 24 keying bits (depending on the number of tables).

Next, with this simplified system, at first it seems as though the ability to cancel layers should give great power to develop powerful relationships between columns. But after having done this, it is not clear where to go from there.

This is not a successful attack, because, at this point, it does not provide insight into any particular table value, or even build relationships between particular tables in different columns.

Acknowledgments

David Wagner was kind enough to look at the previous version of these arguments, and to voice concerns about the ability to cancel the right-going diffusions.

Terry Ritter, his *current address*, and his *top page*.

Last updated: 1998-03-17

Variable Size Block Ciphers

Designs for Achieving Dynamically Variable Block Size by Byte

Ciphers formed from layers and columns of small components. Each (typically byte-width) column is similar, so such a cipher can be extended, byte-by-byte, to arbitrary size.

Layers may perform confusion (such as an array of keyed substitution tables), typically one-way diffusion (such as an exclusive-OR chain), or both (such as a Latin square chain).

Particular designs of 80 bit (10 byte) and 1600 bit (200 byte) wide ciphers were investigated experimentally. Results indicate that specific designs with four or more confusion layers, intermixed with diffusion layers in both directions, perform like the expected random permutation. In practice, more layers may be required for optimal strength.

Contents

- Overview
- Background for Examples
- Example 1: Simple Exclusive-OR Diffusion
- Example 2: Diffusion Across Layers
- Example 3: Balanced Block Mixing for Diffusion
- Example 4: Latin Square Combined Confusion and Diffusion
- Operation
- Keying
- Overall Diffusion
- Support for Analysis
- Design Opportunities
- Conclusions

Overview

Research has produced constructions for practical, efficient, and effective block ciphers which handle data blocks of various size. These constructions are easier to analyze than conventional ciphers. They also support new approaches in dynamic keying and authentication, as well as other features which can only be practical within the context of a variable size block cipher.

Variable Size Block Ciphers have some significant advantages:

- **Fundamental Speed Advantage:** there are relatively few fast and simple layers
- **Fundamental Strength Advantage I:** there is a large amount of keyed internal state
- **Fundamental Strength Advantage II:** a large block size means more uncertainty in each plaintext block
- **Better Fit to Some Existing Systems:** some buffering may be avoided entirely
- **Better Fit in Inherently Variable-Size Uses:** such as voice CODEC's, or a database with variable-size fields
- **Practical Support For a New Form of Keying**
- **Practical Support For a Secure Authentication Code**
- **Inherently Scalable:** supports testable small versions *with the exact same implemented program code*
- **Inherently Supports Easier Analysis:** these constructions use mathematically-simple yet substantial operations.
- **Need Not Expand Data At All**

Background for Examples

Examples 1 through 4 are 80-bit block cipher designs. The 80-bit block was chosen as being larger than the 64-bit block of DES, and not a power-of-2 size (as many ciphering architectures must be). Since each design uses exactly the same block size, their effectiveness can be compared directly.

These ciphers are composed of horizontal rows or layers of similar components, and vertical columns which often contain different component types. The heart of each cipher is the vertical column composed only of operations which are one element in width. In these particular designs, each element is one byte wide; this means that all operations are byte-width operations, and all internal data paths are eight bits wide.

Each column in any one of these ciphers contains the same connections, table-positions and operations. The particular tables used in each column generally will vary. The columns differ in the particular adjacent columns (if any) to which their data and diffusion inputs and outputs are connected. Each cipher can be extended to arbitrary size by adding more columns of virtually identical structure. Dynamically variable size operation generally requires a dynamic selection among keyed substitutions for each particular substitution position.

Since diffusion is necessary across the width of the block, each column communicates with adjacent columns. This means that the leftmost and rightmost columns will produce diffusion carry outputs (which can be ignored) and will require external input values.

These constructions use simple invertible diffusion and confusion layers (or combined layers). Since each of the layers is separately invertible, the cipher itself is invertible, so that enciphered data may be deciphered. Deciphering occurs by producing the inverse of each layer and processing each layer in reverse order from that used in enciphering.

These ciphers are primarily keyed by permuting each of the substitution tables under the control of a keyed cryptographic RNG. Dynamically-variable block size operation generally requires selecting the particular substitutions used in each position from a heap of keyed substitutions. Additional methods for low or zero-overhead dynamic keying are also available.

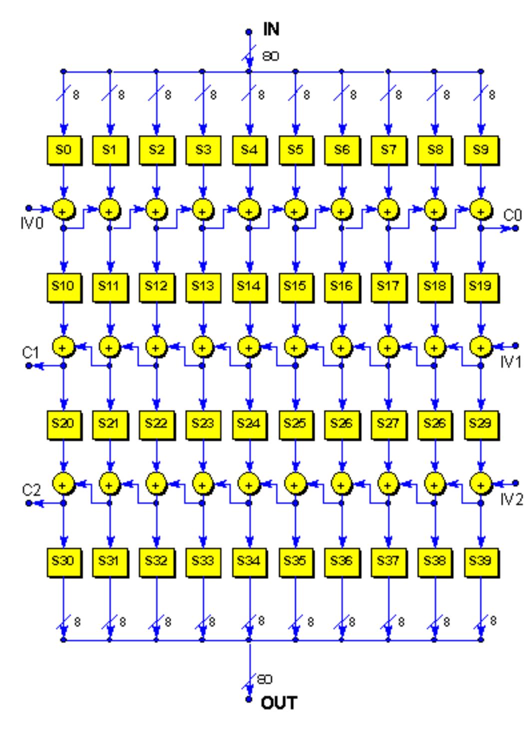
The example figures are data-flow diagrams. That is, the figures represent the flow of data from some source through a data path to an operation, the operation itself, and the flow of the result through another data path to some data sink or user of the result value. In a hardware realization, a data path might be actual metallic connections. In a general-purpose computer realization, a data path might simply represent the movement of data from one operation to another.

Example 1: Simple Exclusive-OR Diffusion

Example 1 is an 80-bit block cipher built solely from variable size layers. This makes the cipher easily extendible (in byte-by-byte steps) to arbitrary size, either at design-time, or dynamically during operation.

Example 1 includes, in alternation, four layers of substitution operations and three layers of exclusive-OR operations. Like all of these variable size block cipher designs, the structure of each column is the same, although the actual substitution tables used will vary.

The 80-bit input block is split into ten 8-bit columns and processed by a confusion layer. All of these results are mixed by an exclusive-OR chain in a right-going diffusion layer. Subsequent diffusion layers are left-going. Each diffusion layer implies both an Initial Value from outside the cipher and a Carry value which can be used for expansion. In practice, yet another pair of layers might be needed for best strength.



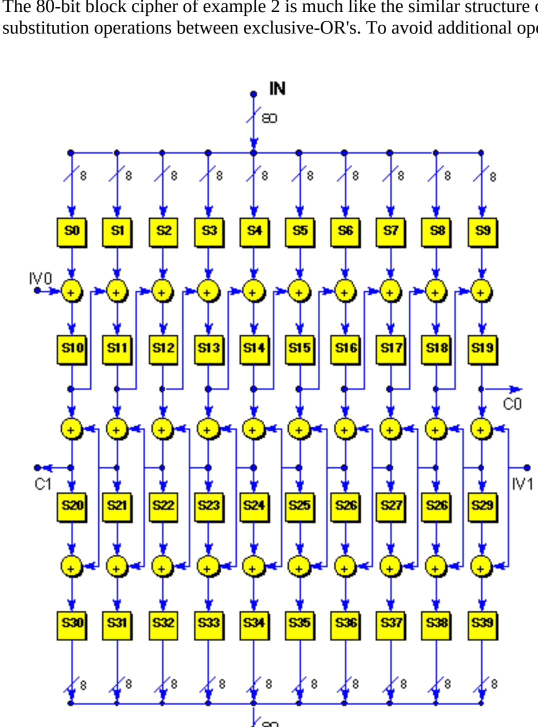
To understand the purpose of the top three layers, it is useful to consider the operation in the case of two different input blocks: first, some random block, and second, the same block with only the leftmost bit changed. The different or changed bit produces a different value on the leftmost input substitution. Thus, at least one bit will change on the substitution output, and will be propagated across the full width of the block through the first diffusion layer. This will change the value into each of the substitutions in the second confusion layer, thus changing the result from each substitution. In this way, a change in any one input bit affects the column which includes that bit, and also each column to the right.

It is at least possible that two input changes could cancel, to produce no diffusion effect on subsequent substitution tables on one layer. While this appears to have little statistical impact, it could certainly be avoided with an additional set of confusion and diffusion layers with diffusion in the same direction.

The remaining confusion and diffusion layers have diffusion in the opposite direction to produce the overall diffusion results expected from a block cipher.

Example 2: Diffusion Across Layers

The 80-bit block cipher of example 2 is much like the similar structure of example 1, with improvements in diffusion. It is found by experiment that diffusion improves when a diffusion layer includes substitution operations between exclusive-OR's. To avoid additional operations or components, the diffusion path is taken from the result of the next confusion layer.



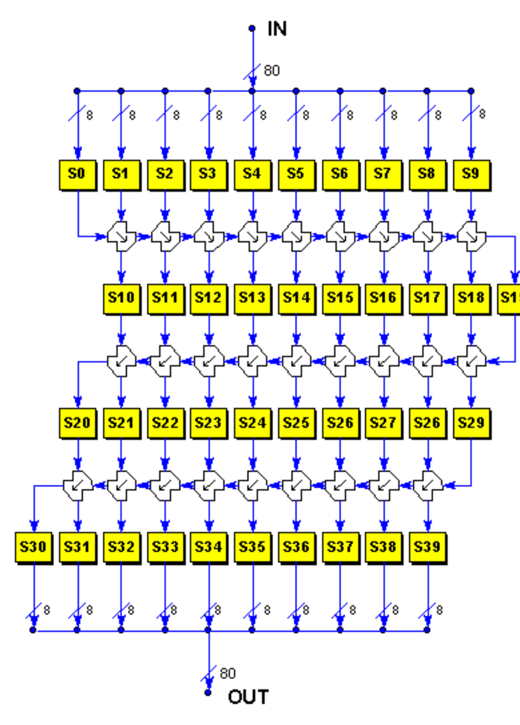
Unlike example 1, some of the diffusion paths used in example 2 include intermediate substitutions using substitutions already in place. Clearly, it would be possible to introduce new substitution operations between each exclusive-OR in the diffusion paths, but much of the benefit of such a structure is achieved here without added computation.

To decipher the block, again note that each layer is separately invertible. We simply produce the inverse tables of those used for enciphering, and process the block "up," from OUT to IN in example 2. The direction of diffusion in each diffusion layer remains unchanged.

Example 3: Balanced Block Mixing for Diffusion

The 80-bit block cipher of example 3 exploits the reversible two-port nature of Balanced Block Mixing. This provides somewhat better diffusion than exclusive-OR, and avoids the need for external Initial Values, but does require a little more computation.

The byte-width Balanced Block Mixing operations each take two input bytes, and produce two output bytes, in the direction indicated by the internal arrow.



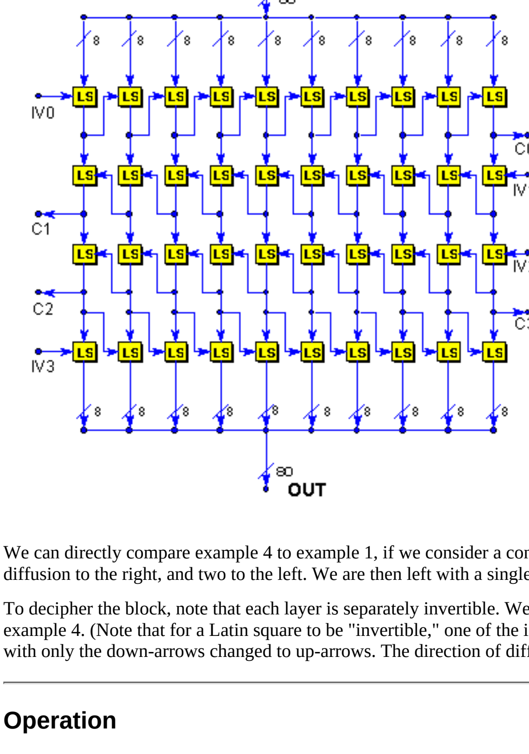
Because the structure of the mixing component differs from the exclusive-OR used in examples 1 and 2, the diffusion architecture of example 3 is accordingly different. In particular, there is always one less Balanced Block Mixing in the diffusion layers than there are element-wide data paths and substitution elements in the confusion layers. We can think of the Balanced Block Mixing as fitting in the space between adjacent confusion columns.

To decipher the block we simply produce the inverse confusion tables of those used for enciphering, and process the block "up," or from OUT to IN in example 3. That is, a data-flow diagram for deciphering is the same as example 3 with all arrows reversed. The Balanced Block Mixer outputs are made inputs, and the inputs outputs, and diffusion occurs in the opposite direction.

Example 4: Latin Square Combined Confusion and Diffusion

The 80-bit block cipher of example 4 is unique in that, of the four examples, it creates a cipher from a single component type. That component type is the Latin square combiner, which simultaneously provides both confusion and diffusion.

In a byte-width Latin square, each of 256 rows and 256 columns will have exactly one occurrence of each byte value 0..255. Before operation, the needed Latin square (or squares) are constructed under the control of one or more cryptographic keys. Each operation in this example might well use the same keyed Latin square.



We can directly compare example 4 to example 1, if we consider a confusion layer and diffusion layer in example 1 as a single layer in example 4. In both cases, we first have one confusion and diffusion to the right, and two to the left. We are then left with a single confusion layer at the bottom of example 1, and a different sort of isolation layer in example 4.

To decipher the block, note that each layer is separately invertible. We simply produce the appropriate "inverse" square from that used for enciphering, and process the block "up," from OUT to IN in example 4. (Note that for a Latin square to be "invertible," one of the input values must be known, as is the case in this example.) A data-flow diagram for deciphering is exactly the same as example 4 with only the down-arrows changed to up-arrows. The direction of diffusion in each diffusion layer remains unchanged.

Operation

Before operation, all the various substitution tables to be used are initialized and shuffled under the control of a key, and any Initial Values will also be selected. In general, each table will be a unique ordering of the same values.

When substitution operations are indicated, the particular table used at each position could be fixed at design time, or selected dynamically, during operation, from among many such tables.

Note that strong block ciphering generally requires a block size of eight bytes or more. While the exact same VSBC structure could operate on smaller blocks, small blocks just are not strong cipherings. Thus, the external system should apply a dynamic VSBC in ways which will assure that each block has at least minimum size, if this is at all possible.

In all cases, deciphering is performed by very similar constructions which process data from the bottom to the top. In most cases, the diagram for deciphering is exactly the same as that for enciphering, with all downward data paths replaced by upward data paths.

Normally, with exclusive-OR or Latin square diffusion, the direction of diffusion does not change when deciphering. However, when Balanced Block Mixers are used, for each mixer, both outputs become inputs, and both inputs become outputs, meaning that the diffusion direction is also reversed.

When deciphering, each substitution table is replaced by its inverse, and each Latin square is replaced by its appropriate inverse. If the Balanced Block Mixers are keyed, each of those will be replaced by its inverse; like exclusive-OR, the usual Balanced Block Mixer is its own inverse.

Keying

The confusion layers provide primary keying. In a typical confusion layer, each element of the data block is transformed by an associated substitution operation. The substitutions are "keyed" or arbitrary selections from among all 256 factorial possible substitutions.

We can support a huge keyspace by using a key of reasonable size to initialize the state of a random number generator. The sequence of values from the random number generator mechanism can shuffle all of the tables in the cipher. Both the design of random number generators and their use to shuffle invertible substitutions are well known and are not improved by the VSBC innovation.

Some dynamic keying may be provided by diffusion-layer inputs (these are the start of the diffusion channels on one or the other side of the block). More dynamic keying can be added by including a keying layer in the cipher. This layer would simply combine a key of the block width with the data. Combining could be additive, Latin square, or other cryptographic combining.

In a dynamically-variable size cipher, a key of any width can be produced by a keyed random number generator. While this would give us a sort of "stream cipher" layer, the sequence would "re-originate" for each block. The VSBC structure would provide overall diffusion which is not available in a stream cipher.

Additional dynamic keying can be provided by including a key as part of the data block. This would produce more ciphertext than message (enciphering would appear to expand the data), but the resulting key would be reproduced by deciphering, and could be used to authenticate the original data. Since, in a variable size block cipher, the data block can be as large as desired, even an 80-bit dynamic key could be a minor overhead. And it might well provide an authentication which may be necessary anyway.

Overall Diffusion

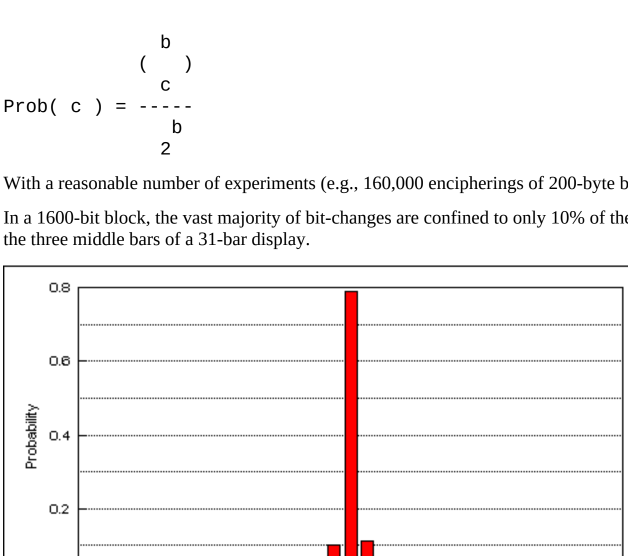
In these designs, overall diffusion is produced by separate layers diffusing in opposite directions. Similar results can be produced by layers diffusing in the same direction, provided the carry output from an early layer is communicated to the initial value input of one (or more) later layers. The latter approach would have to be taken carefully, however, since it could circumvent the isolation of intermediate confusion layers, and might be a weakness which could be attacked.

The overall diffusion characteristics for a block cipher are relatively easy to check by experiment: For a random key we choose a random data block and encipher it. Then we change a bit in the original block, encipher that, and compare the result to the original ciphertext, counting the number of bits which changed. We do this for a large number of random blocks and a large number of keys, producing a distribution which is the number of occurrences of each possible number of bit-changes. We know what to expect because the probability of finding any particular number of bit changes c in a block of b bits is:

```
               b
             (   )
               c
Prob( c ) = -----
                b
               2
```

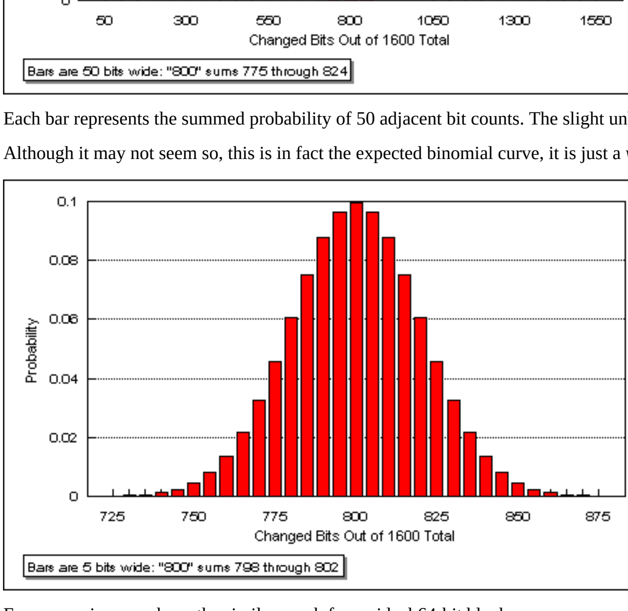
With a reasonable number of experiments (e.g., 160,000 encipherings of 200-byte blocks) large blocks produce an especially clear and distinct diffusion signature.

In a 1600-bit block, the vast majority of bit-changes are confined to only 10% of the possible range. In fact, 99.98% of the time, the observed bit-changes should fall in the range of 725..874. This is just the three middle bars of a 31-bar display.

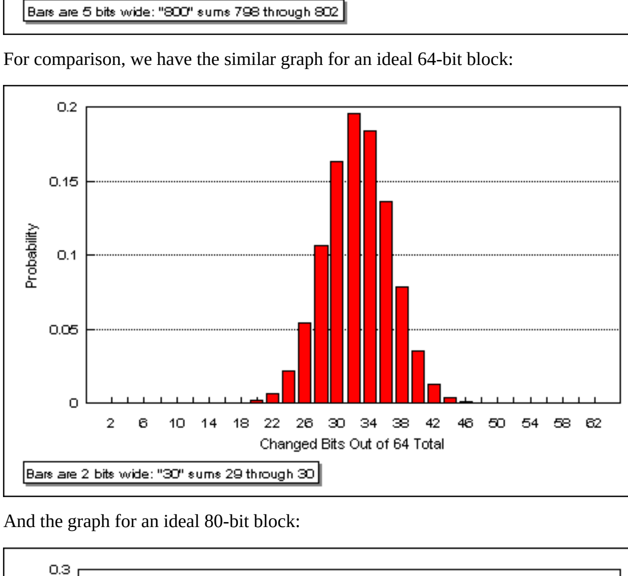


Each bar represents the summed probability of 50 adjacent bit counts. The slight unbalance in the two sides is an artifact of summing an even number of bits per bar.

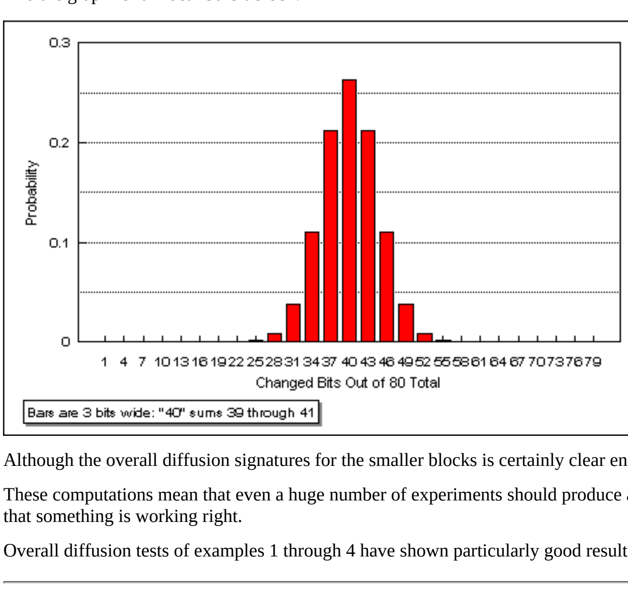
Although it may not seem so, this is in fact the expected binomial curve, it is just a *very narrow* curve.



For comparison, we have the similar graph for an ideal 64-bit block:



And the graph for an ideal 80-bit block:



Although the overall diffusion signatures for the smaller blocks is certainly clear enough for testing, the diffusion signature is clearer in the larger blocks.

These computations mean that even a huge number of experiments should produce a very narrow range of expected values. Finding this sort of experimental distribution is a clear and obvious indication that something is working right.

Overall diffusion tests of examples 1 through 4 have shown particularly good results, sometimes surprisingly better than very similar architectures.

Support for Analysis

These variable size block cipher examples all use operations which have a relatively-simple mathematical description and so can be well understood and manipulated. Despite this simplicity, the table structures are relatively-large and so contain a large amount of arbitrary internal state.

For example, all of the substitution tables in these variable size block ciphers are just different arrangements or permutations of the same values. Among other things, this means that we can guarantee that any input change *must* select a different output value, and thus produce an output change. More general confusion functions, such as those supported in Feistel architectures, do not have this property and do not support this sort of reasoning unless limited to particular functions.

Similarly, the use of layers which provide diffusion across the width of the block simplifies diffusion questions compared to other ciphers. For example, most conventional Feistel designs apparently find the number of rounds needed to for complete diffusion by experiment. In the variable size block cipher examples given here, we can do the same sort of experiments, but also can reason about the effect of each separate layer, and there are relatively few layers to analyze.

Since there is no way to measure or guarantee the strength of any practical cipher, any improvement in the ability to reason about the mechanism is a big advantage.

Design Opportunities

- **High Speed Operation:** These Variable Size Block Cipher constructs are fast ciphering mechanisms, far faster than DES.
- **Increased Design Flexibility:** The variable-size feature is a better interface to many other systems, such as multiple or variable-size database fields and perhaps also voice CODECs. Variable size blocks can avoid data buffering. Disk sectors can be directly ciphered as a single block. And the very same design might be used as a conventional 64-bit block cipher. Moreover, entire layers are easily added or removed to vary the strength-to-speed tradeoff, without complete re-design.
- **Use Modern VLSI Strengths:** Variable Size Block Cipher constructs tend to be architecturally cleaner than other other ciphering mechanisms. VSBC's make extensive use of storage structures which are themselves extremely regular, and make better use of silicon real estate than irregular structures. A fixed 80-bit-wide cipher like that in Example 1 uses 40 byte-wide substitution tables, for a total confusion store of 10KB. This is under 2 percent of the store available in a single common 4Mb dynamic RAM chip. A fixed-width hardware design using these same techniques could perform each substitution in a layer in parallel. Moreover, the layers could be pipelined, for speed limited mainly by diffusion chaining. For ultimate speed, chaining delays can be controlled by using smaller blocks.
- **Built-In Scalability for Production Testing:** A dynamically-variable VSBC design can be extensively tested as a small (two- or three-element) cipher using *exactly the same code (or hardware)* as used in production ciphering.
- **Built-In Theoretical Testability:** Variable Size Block Cipher designs are easily made with 2-bit, 3-bit, or 4-bit element sizes. When used as a block cipher of from 2..4 elements, the transformations produced by such designs can be *fully explored*, in massive fundamental contrast to other cipher architectures. This should lead to far greater insight into the functioning of the cipher and its strength, and lead also to solid predictions for the production versions which are necessarily far too large to fully explore.

Conclusions

Various novel constructions for block ciphering are presented. Perhaps most importantly, these constructs can be induced to cipher blocks of dynamically-variable size. This forms a new ciphering component -- the Variable Size Block Cipher -- which itself enables new ciphering architectures and new potential benefits.

These constructions are fast, efficient, scalable, and are relatively easy to work with in analysis.

This is the original public exposition of this sort of ciphering construct. Further research may indicate changes in the given constructions. But the general concept that Variable Size Block Ciphers can exist, that some efficient realizations do exist, is itself a significant step.

Terry Ritter, his *current address*, and his *top page*.

Last updated: 1995-11-21

VSBC Newsgroup Discussion

Variable Size Block Ciphers discussed in sci.crypt.

The goal is a block-cipher architecture which can have an essentially arbitrary and dynamically-variable block size. It is necessary that good diffusion be produced from all plaintext input bits to all ciphertext output bits. It is desirable that a fixed number of processing layers evenly diffuse blocks of any size, or else there would be a strong motive to use small blocks.

Contents

- The Original Announcement
- Announcing Realized Prototypes

The Original Announcement

- 1995-08-20 Terry Ritter: The original announcement.
- 1995-08-21 Ross Anderson: "Two such ciphers appeared in 1993 - WAKE by David Wheeler and a proposal from Burt Kaliski and Matt Robshaw. They are both in `Fast Software Encryption', Springer LNCS 809"
- 1995-08-23 Terry Ritter responds to Ross: "While Kaliski-Robshaw does handle large 1 KB blocks [...] this is a particular design for a particular (fixed) size block. WAKE is an autokey stream cipher. In a stream cipher, diffusion can occur only to that part of the stream *following* a particular datum."
- 1995-08-24 Ralf Brown: "And I proposed another approach to variable-size blocks, namely using a Feistel network and "sliding" it along the input, back in April."
- 1995-04-02 Ralf Brown. Ralf's previous message. (Note the absolute lack of any concept of dynamically *variable* size, such as size parameterization or the like.)
- 1995-08-25 Terry Ritter responds to Ralf: The mentioned ciphers differ from "Variable Size Block Ciphers" as defined.
- 1995-08-25 David Wagner: "No go: this is easily cryptanalyzed by differential cryptanalysis." (This posted response mistook the design as using but a single table in each row, but later private e-mail did show how the real design could be attacked.)
- 1995-08-25 John Kelsey: "...it seems odd to me that you don't need more rounds to handle larger blocks." Also detailed questions and comments.
- 1995-08-26 Paul Rubin: "Isn't RC5 a variable width block cipher, sort of?"
- 1995-08-26 Terry Ritter responds to David: "each substitution is intended to be a separate keyed (shuffled) table. [...] Currently, I am less interested in strength than overall diffusion. My point is that it seems amazing -- wondrous -- that an overall bit-level diffusion effect can be generated for an essentially arbitrary block width by a fixed-depth structure."
- 1995-08-27 Ralf Brown responds to Terry: "I wasn't thinking of the above, but an extension thereof which I posted to sci.crypt.research at the end of April [...]"
- 1995-05-01 Ralf Brown: Ralf's other previous message. (This message does say "effectively unlimited block size", but there is absolutely **no** discussion of a dynamically *variable* block size. This design also does not diffuse evenly over the whole block -- the first and last elements get less diffusion -- and needs more layers to process larger blocks.)
- 1995-08-27 Ralf Brown responds to Paul: "Parameterized. You can set various sizes beforehand, to get a different variation of the cipher."
- 1995-08-27 Terry Ritter responds to John's detailed questions and comments.

The technical criticism to these brand-new structures comes from David Wagner, and his "No go" response certainly sounds ominous. It took me a long time to understand this criticism and place it in context, even with several other messages from David by private e-mail.

As I understand it, David comments that if we change adjacent input bytes, we can match values in the top-level substitutions, and when this is repeated, it essentially solves that confusion layer. Although I was aware of the first part of this, I did not see how it would lead to success. Thanks David!

Thus, what I had seen as a worst-case block cipher test (the single-bit-change avalanche results) ignores the important possibility of correlations in multi-bit changes. (I expect that we could pick this up by trying all 64K values of two adjacent bytes over multiple keyings.)

But David himself comments that we can correct the problem in the cipher simply by adding another right-going diffusion layer to the original structure. So the "No go" response is not a blanket indictment of the technology, but is instead a good insight about ways in which these structures can be weak. We have every motive to reduce the number of layers, but we can easily go too far. Don't do that.

Announcing Realized Prototypes

- 1996-02-11 Terry Ritter: Five realized prototypes and comparative speed measures. (13K)

Terry Ritter, his current address, and his top page.

Last updated: 1996-02-15

Cryptography: Is Staying with the Herd Really Best?

Terry Ritter, Ritter Software Engineering

A recent Internet Watch column argues that new cryptography is bad cryptography. Drawing an analogy to the medical profession, it says, "A good doctor won't treat a bacterial infection with a medicine he just invented when proven antibiotics are available." That certainly sounds reasonable. But, in a very real and practical sense, there *is* no proven cryptography. And this is not just an issue of mathematical proof: The cryptographic profession simply can't tell whether or not a cipher really is protecting data. It is as though medical doctors were telling us about their cures when in reality they couldn't even tell if their patients were alive or dead.

It is not that we want to avoid cryptanalysis; indeed, we want all the analysis we can get. And it is true that a brand new cipher has had scant time for analysis. But the result of even deep analysis is not a proven design; it is just something that we don't positively know to be weak. This slight shift of meaning is the basis for understanding what cryptography can and can't do. For one thing, it means that any cipher—no matter how deeply analyzed—could be weak in practice. And that means that anyone concerned with real security probably should consider using something other than the same cipher as everyone else. One possibility is using new cryptography in new ways, which is the exact opposite of what that previous column suggests.

There is neither proof nor a test of overall strength for either new or old cryptosystems.

Surely, we all would like to have a fully reviewed library or cipher in the same way that we would like to have a fully debugged program. But not even lengthy review or analysis guarantees either cryptographic strength (the ability to resist attack) or a lack of program bugs. For example, most crypto experts probably would agree that just because 20 years of analysis of the US Data Encryption Standard has not found an easy break doesn't mean that no easy break exists. And if a break does exist, it may have been actively exploited for years without our knowing. We certainly couldn't call that a strong cipher. In practice, even extensive review is not a rational or scientific indication of strength.

This is not an issue of perfection versus reality, and it isn't like software where we tolerate various bugs and still get real work done. In software, the bugs are generally peripheral to our goals, and we usually know if we are getting what we want. But in cryptography, we have no idea whether or not someone can break our cipher, even if there are no bugs at all in the program.

CONFIDENCE IN CIPHERS

Perhaps the central problem in cryptography is how we can have confidence in a cryptographic design. Ways often mentioned to gain confidence in ciphers include mathematical proof, practical tests, open cryptanalysis, and long use.

Mathematical proof and practical tests

Despite more than 50 years of mathematical cryptography, there is no complete mathematical proof of strength for any practical cipher, at least not in the open literature. (A one-time pad is often assumed to be secure, but is impractical in most cases.)

Likewise, there is no set of tests that measures all possible weaknesses in a cipher. The very feature we need—strength against unknown attack—is something we can't measure. This is like a publisher who can measure the quality of paper, printing, and binding yet still not know the quality of a book or articles. The essence of a cipher is not the measurable ciphering process itself, but rather the effect that process has on confounding each opponent. Cipher quality is necessarily contextual, and we can't know the context.

Editor: Ron Vetter, University of North Carolina at Wilmington, Department of Computer Science, 601 South College Rd., Wilmington, NC 28403; voice (910) 962-3667, fax (910) 962-7107; vetterr@uncwil.edu.

Cryptanalysis

Cryptanalysis is the art of trying to find an easy way around a security design's cryptographic protections. While many specific attacks are known in an expanding literature, the number of possibilities is essentially unbounded. There is no exhaustive theory of cryptanalysis. Without such a theory, cryptanalysis that does not find a problem does not testify that no problems exist. Cryptanalysis gives us an upper bound for strength, but not the lower bound that describes the minimum effort needed to break a cipher.

Nor does cryptanalysis provide evidence that our cipher is strong. Surely we use only ciphers we can't break. But predicting what an opponent can do on the basis of what we can do is at the very essence of weak cryptography. The classic examples occurred in Germany and Japan in World War II, but every broken system is a failed assumption of strength. We can either learn from history or repeat it on the losing side.

Long use

Our opponents operate in secrecy and do not reveal their successes. If they break our cipher and take some care to protect the results, we will continue to use that broken cipher, all the while assuming our data is protected. Confidence from long use is a self-delusion that springs from not specifically being told that our cipher has failed. We hope that our lack of knowledge means that our cipher has not been broken. But if hope were enough, we wouldn't need cryptography.

THE CRISIS OF CIPHER CONFIDENCE

There is no known proof, measurement, or analysis that provides confidence in cipher strength. Cryptosystems both new and old are in exactly the same boat: Against unknown attack, even an extensively reviewed system may not be as strong as one not reviewed at all. An implied problem with a new cryptosystem is that we can't know that it is strong. But the real problem is that we can't know that the old system is strong—and that is the system we are actually using.

If academics refuse to address patented cipher designs on a rational, technical basis, they won't develop the background to understand or compare the new cryptographic technologies. It is even possible that there may be a practical security advantage to a patented cipher: Since no one can prove that any cipher is secure, absolute confidence is simply not available. Any cipher can fail at any time. But if a patented cipher fails, we may be able to prove that someone used an unlicensed deciphering program and take legal steps to recover our losses.

There is no known proof, measurement, or analysis that provides confidence in cipher strength.

WHAT CHOICES DO WE HAVE?

Even if we consider every cipher as possibly insecure, we do have alternatives. Instead of reacting to rumor or waiting for academic breakthroughs, we can proactively use new approaches and new technology. We can, as a matter of course, multicipher our data: We can use one cipher on the plaintext, a different cipher on the resulting ciphertext, and yet another cipher on that result. In general, if even one of the three ciphers is strong, our data is protected. And even if each cipher has a weakness, it may be impossible to exploit those weaknesses in the multiciphering context. For example, multiciphering protects individual ciphers from known plaintext attacks.

Another alternative is to use a wide variety of structurally different ciphers and to randomly select ciphers by automatic negotiation. In addition to terminating any existing break, this spreads our information among many different ciphers, thus reducing the reward for breaking any particular one. Another step is to continually add to the set of ciphers used. This increases costs for our opponents, who must somehow acquire, analyze, and construct software (or even hardware) to break each new cipher. But new ciphers would be only a modest cost for us.

Absent a mathematical theory to assure a high cost of cipher-breaking, experimentation is the main way to test a cipher's strength, but real designs are far too large to know in any depth. So another alternative is to construct scalable designs that produce both tiny toy ciphers (which can be broken and deeply examined experimentally) and large serious ciphers from the same specification.

Despite the frequent cryptography articles in IEEE journals, cryptography is an art, not an engineering discipline: The property we seek to produce—strength against unknown attack—is not measurable, and so it is literally out of control. But if we avoid new technology, we help our opponents, who certainly don't want to deal with a wide array of new ciphers.

Not applying new technology is wrong—wrong for an innovator who seeks compensation, wrong for the user who wants the best systems, and wrong for those who want the field to mature. It is necessary, therefore, to take recommendations against using new cryptography along with a healthy dose of reality. ❖

Terry Ritter is the owner of Ritter Software Engineering. He received a BS in engineering science from the University of Texas at Austin. He is a member of the IEEE Computer Society and the ACM. Contact him at ritter@io.com.

For More About Cryptography

Additional information about related topics can be found on Ritter's Web site (http://www.io.com/~ritter/), including a basic introduction to cryptography (http://www.io.com/~ritter/LEARNING. HTM), an extensive crypto glossary (http://www.io.com/~ritter/ GLOSSARY.HTM), literature surveys, Usenet conversations, crypto links, and his own work.

Experimental Characterization of Recorded Noise

Statistical and Graphic Views of Noise

Experimental measures reveal problems where previously none were suspected.

A *Ciphers By Ritter* Page

Terry Ritter

This is phase 2 of my investigation of physically-random noise sources. In phase 1, various noise sources were constructed and their outputs recorded into .WAV files by a computer sound card. Here we analyze the collected data to see what we got.

For this phase, I wrote the statistical routines, re-wrote the FFT's, and wrote the graphing routines. The graphs are saved to .BMP files and converted to .GIF for use with HTML. The statistic results are saved as HTML. To minimize display delay, the HTML "table" construct was not used.

Characterizing the noise waveform is a fairly unusual approach to random generator certification. The usual approach is to quickly digitize the noise into supposedly random bits, and then present *those* for randomness checking. I expect that bit-level results tend to hide real problems which could be detected far easier if we had access to noise waveform data.

Noise Data Characterizations

ZENER1.WAV

My first noise recording, using my classic all-transistor noise-generator circuit, updated with a pseudo-zener IC.

ZENER1ZY shows a non-flat frequency response which rolls off about 6dB between 2kHz and 10kHz. The normal curve is also somewhat off. But the autocorrelation graph captures our attention with completely unexpected structure.

ZENER157 covers a larger sample and so resolves the autocorrelation structure even more convincingly. Maybe we can fix it up by modifying the data.

ZENER10T shows that simply subtracting the mean has no effect on the autocorrelation structure.

ZENER1P7 "fixes" the data by subtracting the previous data value. This is differential data. And now we see the autocorrelation structure is gone, and the normal curve is fairly nice, especially with respect to finding some counts in the +/-4th sd.

With the data fix-up, this is probably an acceptable generator.

FM1.WAV

This is the commercial FM receiver source. Actually, it is just a headphone FM radio, in monaural mode, with signal taken across earphone wires, with the earphone still connected. When placed in a metallic conductive tin, we can hear a local station drop into noise as the top is put on. So is FM hiss good noise?

FM11WOWK Here we have a not-great frequency response, starting high and sloping down to the end. And the normal graph is a bit off. But it is the autocorrelation which again captures our attention: we see structure. Is that structure real? To answer this question we can use a somewhat larger sample.

FM1ME904 With more data analyzed, the structure in the autocorrelation graph is even more pronounced. This is not good. Suppose we try to "fix-up" the data.

FM1889TN The data fix-up appears to solve the correlation problem, but does not quite produce a good normal curve.

In the end, this commercial FM receiver is not a great noise generator.

DIG1.WAV

DIG1 is the output from an MM5837 Digital Noise Generator IC. As such, it is not useful as an unknowable randomness generator. But it might provide calibration to our measurements and an example of a good generator.

DIG1PIO8 The raw data shows an extremely flat frequency response curve with a roll-off starting at about 18kHz which may be inherent in the sound card. The autocorrelation shows some bias. The normal curve data seem reasonable, if not extending to the limits. In this case the graph may have been scaled somewhat high because the center data point (which is tested for scaling) is randomly low.

DIG10EKB The frequency response, previously flat, now shows a fixed rising slope. The data fix-up process is thus revealed as a form of digital high-pass filter. The autocorrelation is improved, but the normal curve does not extend much beyond 3 sd's, which seems a little limited.

FM2.WAV

FM2 was recorded as the noise output from an old communications IF strip. This complex circuit presents the opportunity of seeing 10.7MHz noise, translated down to the 455kHz detector, and may be flatter than we might get otherwise. Of course, a lot depends upon the detector.

FM2QXTMB Here we have a frequency response with a small peak about 1kHz, with a nice flat plateau until it rolls of starting about 17kHz. The autocorrelation is nice, perhaps with some bias. But the normal graph is terrible!

FM2JSVH7 Here we enable the data differential fix-up. The frequency response now rolls to a gentle peak around 16kHz, and the autocorrelation is good. But the normal curve is not only bad, it is peculiar.

In the end, we do not get an acceptable result from this generator.

ZTN1.WAV

This is the original ZENER1 circuit, with the second transistor replaced by an audio amplifier IC, which should have better linearity.

ZTN1QOY9 First, the frequency response starts out high and droops down thereafter; this is quite similar to ZENER1. The autocorrelation shows actual structure; although not quite the same as ZENER1, it is similar. And the normal graph is strikingly similar to ZENER1. It thus appears that the second stage has not been much improved.

ZTN1ALIQ A substantially larger sample confirms the autocorrelation structure.

ZTN1YUN6 With data fix-up, we have rolling hill of a response generally peaking from 10kHz to 18kHz or so, which is remarkably similar to the fixed-up ZENER1. The autocorrelation structure is gone, and the normal curve looks fairly good, also like the fixed-up ZENER1.

In general, this generator should be acceptable with data fix-up, although not much of an improvement over the original.

ZTC2.WAV

This is the same circuit as ZTN1, with a capacitor used to increase the gain of the IC output amplifier.

ZTC2L8W2 Here we have a frequency response which gently peaks about 5kHz, and then slopes down evenly to the end of the graph. It may be that the feedback cap value provides a welcome low-frequency cut which sets up the 5kHz peak. Most surprisingly, the autocorrelation we saw in the circuit without the cap is now gone; this may well be due to the new low-frequency cut, and is very welcome. The normal graph looks a little raggedy on the ends, however.

ZTC2BHA2 With "diff from prev" filtering, the frequency response is a broad plateau between 5kHz and 18kHz. The autocorrelation is good, and the normal curve is good if not great (this is another case where the random central value influences the scaling, but that would not affect data on the far tails).

All in all, a reasonable generator.

NNN1.WAV

Now we come to the attempt to use the "zener" IC is voltage mode. This eliminates the first transistor, which should be an advantage, because we expect a modern feedback amplifier to be more linear. The system may require more gain, however.

NNN1QMBF This was intended as a base measure of the 2-stage IC amplifier system, which has noise of its own. We see far too much low-frequency noise, which may be due to normal transistor 1/f noise. At about 2kHz it starts a nice flat plateau extending to about 18kHz, which appears to be a system limit. Autocorrelation shows significant structure, which could be mostly related to massive low-frequency signal. And the normal graph is nice.

ZNN1.WAV

ZNN1 adds the "zener" IC as a voltage noise source into the NN1 amplifier system.

ZNN1CEXP We see a little more range in the sampled values, but otherwise looks fairly similar to the system without the noise source.

ZNN1BGNX Fixing up the data leads us to a nice rising response, peaking about 17kHz and then dropping off somewhat sharply. The autocorrelation is nice, and the normal curve is great. We could do with a somewhat larger range in sampled values, however.

ZCN1.WAV

ZCN1 adds a capacitor to the first amplifier IC to get more output.

ZCN1Z98P We see a very flat frequency response, except for the 18kHz rolloff which is system related. The autocorrelation is clearly biased, but the normal curve looks if not great, then at least decent.

ZCN1I0DG Here we turn on the data-filter and subtract the mean from each sample. This appears to fix the autocorrelation problem. Note that both the frequency response and normal graphs are essentially unchanged by this filter. This is probably as close as we will get to the ideal white noise generator.

ZCN161T8 When we switch to "diff from prev" filtering, the frequency response changes markedly from flat to a hill. But the autocorrelation problem is fixed, and the normal graph is essentially unchanged.

ZCC1.WAV

ZCC1 adds a capacitor to the second IC amplifier stage, to get even more output.

ZCC13Y33 The frequency response seems a little more droopy at the high end. But the autocorrelation is literally off the graph (for the whole graph), and the normal curve indicates serious positive clipping or compression of some sort.

ZCC1S5VH With "diff from prev" filtering, we get a nice hill for a response, and the autocorrelation is back to what we want. But the normal graph still says something is wrong.

ZCN3.WAV

This is basically a re-check that the modified circuit has returned to ZCN1 conditions.

ZCN3JTFR With "diff from prev" filtering, this seems to be a nice generator.

Terry Ritter, his current address, and his top page.

Last updated: 1999-06-22

Orthogonal Latin Squares, Nonlinear Balanced Block Mixers, and Mixing Ciphers

A *Ciphers By Ritter* Page

Terry Ritter

Block ciphers generally must mix each plaintext bit into each and every ciphertext bit, a result which is commonly called "diffusion." Most modern designs use a Feistel structure, in which diffusion is probabilistic and extended across the block by a series of repeated applications or "rounds." An alternative is to use small mixings, in FFT-like patterns, so each input is *guaranteed* to affect each output across the full width of the block, no matter how wide the block may be. The small mixings should be balanced, invertible, simple, and fit the "butterfly" model which is convenient for the FFT. This mixing is available in the orthogonal Latin squares of a Balanced Block Mixer or BBM.

The application of BBM's to invertible ciphering has been known since March 1994. From the beginning, these designs have used *linear* BBM's, but *non* linear BBM's also exist, and they can be dropped into the old designs for new cryptographic strength. Nonlinear BBM's can be constructed in a "checkerboard" process similar to that used to construct nonlinear Latin squares. A single 256-byte table can hold two orthogonal Latin squares of order 16, and is suitable for mixing 4-bit "nybbles." Two such tables, perhaps dynamically selected from an array, produce an 8-bit-wide mixing. And 8-bit mixings repeated in FFT-like patterns can mix entire blocks of huge size with practical effort and equal effect from every input.

These Mixing ciphers are scalable to small block size, and so can be investigated experimentally in ways most Feistel ciphers cannot. Hardware realizations consist almost entirely of tables, leading to extremely dense chips, high performance, and low-risk development. In software, the cipher block size can vary dynamically at ciphering time in "power-of-2" steps, thus supporting legacy 64-bit blocks, modern 128-bit blocks, and independent 64-byte blocks in the exact same unchanged program.

Contents

- 1998-09-22 Terry Ritter: Orthogonal Latin Squares, Nonlinear BBM's, the original article.

From: ritter@io.com (Terry Ritter)
Newsgroups: sci.crypt
Subject: Orthogonal Latin Squares, Nonlinear Balanced Block Mixers
Date: Tue, 22 Sep 1998 20:12:22 GMT
Lines: 658
Message-ID: <36080427.17744393@news.io.com>

ORTHOGONAL LATIN SQUARES, NONLINEAR BALANCED BLOCK MIXERS, AND MIXING CIPHERS

Terry Ritter   ritter@io.com
Ritter Software Engineering
http://www.io.com/~ritter/

1998-09-22

ABSTRACT

Block ciphers generally must mix each plaintext bit into each and every ciphertext bit, a result which is commonly called "diffusion." Most modern designs use a Feistel structure, in which diffusion is probabilistic and extended across the block by a series of repeated applications or "rounds." An alternative is to use small mixings, in FFT-like patterns, so each input is *guaranteed* to affect each output across the full width of the block, no matter how wide the block may be. The small mixings should be balanced, invertible, simple, and fit the "butterfly" model which is convenient for the FFT. This mixing is available in the orthogonal Latin squares of a Balanced Block Mixer or BBM.

The application of BBM's to invertible ciphering has been known since March 1994. From the beginning, these designs have used *linear* BBM's, but *non* linear BBM's also exist, and they can be dropped into the old designs for new cryptographic strength. Nonlinear BBM's can be constructed in a "checkerboard" process similar to that used to construct nonlinear Latin squares. A single 256-byte table can hold two orthogonal Latin squares of order 16, and is suitable for mixing 4-bit "nybbles." Two such tables, perhaps dynamically selected from an array, produce an 8-bit-wide mixing. And 8-bit mixings repeated in FFT-like patterns can mix entire blocks of huge size with practical effort and equal effect from every input.

These Mixing ciphers are scalable to small block size, and so can be investigated experimentally in ways most Feistel ciphers cannot. Hardware realizations consist almost entirely of tables, leading to extremely dense chips, high performance, and low-risk development. In software, the cipher block size can vary dynamically at ciphering time in "power-of-2" steps, thus supporting legacy 64-bit blocks, modern 128-bit blocks, and independent 64-byte blocks in the exact same unchanged program.

BALANCED BLOCK MIXING

In early 1994 I had the honor to introduce to cryptography an alternative to the Feistel structure for mixing in block ciphers, which I now call "Balanced Block Mixing." (See the original article:

http://www.io.com/~ritter/NEWS/94031301.HTM

and the modern summary:

http://www.io.com/~ritter/BBM.HTM).

These mixing structures typically take two values, mix them, and produce two values as a result.

A Balanced Block Mixer is an m-input-port m-output-port mechanism with various properties:

1. The overall mapping is one-to-one and invertible: Every possible input value (over all ports) to the mixer produces a different output value (including all ports), and every possible output value is produced by a different input value;

2. Each output port is a function of every input port;

3. Any change to any one of the input ports will produce a change to every output port;

4. Stepping any one input port through all possible values (while keeping the other input ports fixed) will step every output port through all possible values.

A Feistel structure may have similar guarantees, provided the "f" function is an invertible substitution. But the "f" function is half a block wide, compared to byte-wide BBM functions expanded by FFT-like mixing patterns. And in a Feistel structure, for any one round, both the input to "f" and the changes produced by "f" are exposed in the output; in contrast, BBM's hide their input values. Feistel structures often use a fixed and known "f" function, which is thus open to extensive analysis, whereas Mixing designs generally use keyed tables and now keyed BBM's as well.

BBM's have been described in various ciphering constructions. (See the articles at:

http://www.io.com/~ritter/CRYPHTML.HTM#BBMTech).

The previous work has assumed one or another form of *linear* Balanced Block Mixing. But it is possible to construct *non*linear BBM's with these *same* *mixing* *properties*. This means that nonlinear BBM's can directly replace linear BBM's in earlier Mixing cipher constructions (provided attention is paid to mixing order to allow reversibility).

A TYPICAL MIXING CIPHER

In figure 1 we have a typical Mixing cipher in schematic form, with 3 "fencing" layers of 8 invertible substitutions each, and two full "FFT-style" mixings between them. If these are 8-bit substitutions, we have a 64-bit block cipher. Each substitution (and now each mixing operation also) is individually keyed.

The vertical dotted lines represent typically 8-bit-wide data paths, and the data flow is from top to bottom. Each S is a substitution table, and *--* represents the "butterfly" action of a single BBM. For 8 elements we have log2(8) = 3 mixing FFT sublayers (Mix0, Mix1, Mix2) of 8/2 = 4 BBM operations each. All BBM's in the same sublayer are independent and can occur simultaneously in parallel hardware. The wider the block, the more BBM's needed, which also means that more computation can occur simultaneously.

```
|   A 64-Bit Mixing Cipher    Fig. 1
|
|    :   :   :   :   :   :   :   :   <- Input Block
|    S   S   S   S   S   S   S   S   <- Fencing
|    :   :   :   :   :   :   :   :
|    *---*   *---*   *---*   *---*    Mix0 0..0, 0..3
|    :   :   :   :   :   :   :   :
|    *-------*   :   *-------*   :    Mix1 0..0, 0..1
|    :   *-------*   :   *-------*    Mix1 1..1, 0..1
|    :   :   :   :   :   :   :   :
|    *---------------*   :   :   :    Mix2 0..0, 0..0
|    :   *---------------*   :   :    Mix2 1..1, 0..0
|    :   :   *---------------*   :    Mix2 2..2, 0..0
|    :   :   :   *---------------*    Mix2 3..3, 0..0
|    :   :   :   :   :   :   :   :
|    S   S   S   S   S   S   S   S   <- Fencing
|    :   :   :   :   :   :   :   :
|    *---------------*   :   :   :
|    :   *---------------*   :   :
|    :   :   *---------------*   :
|    :   :   :   *---------------*
|    :   :   :   :   :   :   :   :
|    *-------*   :   *-------*   :
|    :   *-------*   :   *-------*
|    :   :   :   :   :   :   :   :
|    *---*   *---*   *---*   *---*
|    :   :   :   :   :   :   :   :
|    S   S   S   S   S   S   S   S   <- Fencing
|    :   :   :   :   :   :   :   :   <- Output Block
```

Each of the mixing BBM's uses an orthogonal pair of Latin squares. If these are nonlinear, we must of course process the FFT-style "sublayers" in reverse order when deciphering. So if we use the opposite orders in the top and bottom mixing sections, we can use the exact same structure for both enciphering and deciphering; only the table contents then need be changed.

While Mixing cipher interconnections can seem similar to substitution-permutation (S-P) designs, Mixing ciphers propagate full-width diffusion from and to every component instead of using one-bit connections iterated over multiple rounds. Because mixing designs have diffusion guarantees, iterative rounds are neither used nor needed. An entire Mixing cipher may consist of tables, with no random logic, no exclusive-OR's, and no "rounds" at all. (In practice, we would need multiplexers and write timing to load the tables from external data, then read them back for testing, but speed would not be an issue in those data paths.)

LATIN SQUARES

A "Latin square" of "order n" is an n x n array of n distinct symbols, in which each symbol occurs exactly once in each row and also exactly once in each column, as shown in figure 2.

```
|   The Standard Latin Squares of Order 4    Fig. 2
|
|  a) 0 1 2 3   b) 0 1 2 3   c) 0 1 2 3   d) 0 1 2 3
|     1 2 3 0      1 3 0 2      1 0 3 2      1 0 3 2
|     2 3 0 1      2 0 3 1      2 3 1 0      2 3 0 1
|     3 0 1 2      3 2 1 0      3 2 0 1      3 2 1 0
```

Each Latin square can be seen as a combiner of row and column values, and as a generalization of exclusive-OR. (For more background, see the previous article:

http://www.io.com/~ritter/ARTS/PRACTLAT.HTM

or the Latin square literature survey:

http://www.io.com/~ritter/RES/LATSQ.HTM).

ORTHOGONAL LATIN SQUARES

Two Latin squares can be superimposed so that each element consists of an ordered pair of symbols, one from each square. If no ordered pair occurs twice, the squares are "orthogonal."

```
|   An Othogonal Pair of Order 4    Fig. 3
|
|      0 1 2 3     0 1 2 3       00 11 22 33
|      1 0 3 2     3 2 1 0   =   13 02 31 20
|      2 3 0 1     1 0 3 2       21 30 03 12
|      3 2 1 0     2 3 0 1       32 23 10 01
```

In orthogonal Latin squares, each of the n*n possible ordered pairs of symbols occurs exactly once. So if we know a particular symbol pair, we can identify the row and column to which they belong, and so reverse the transformation. There are no orthogonals of order 2, so there can be no exclusive-OR analogy to orthogonal combining.

There are exactly 576 Latin squares of order 4, so there are 576 * 576 or 331,776 possible pairs which might be orthogonal. Of these, only 6912 pairs actually are orthogonal. If we then reduce every square of every orthogonal pair to standard form, we find that they all reduce to exactly the same standard square, the one shown in figure 4.

```
|   The Standard Square for All Order 4 Orthogonals    Fig. 4
|
|  d) 0 1 2 3
|     1 0 3 2
|     2 3 0 1
|     3 2 1 0
```

This order-4 standard square generates 144 permuted squares, and each of these has exactly 48 orthogonal partners. So if we take one square at random, the probability of the next random square being orthogonal to the first is exactly 1/3.

One way to construct an orthogonal pair at order 4 is to first build a randomized Latin square, by starting with standard square (d) and shuffling rows, columns, and symbols. Then we build other squares repeatedly, in the same way, until some pair is orthogonal.

ORTHOGONAL CHECKERBOARD CONSTRUCTION

The earlier article, "Practical Latin Square Combiners,"

(http://www.io.com/~ritter/ARTS/PRACTLAT.HTM)

described a "checkerboard" construction for large nonlinear Latin squares. Basically, we start with some square and replace each symbol with a full Latin square. We use different symbol sets for these small squares and so produce a valid larger Latin square.

A similar process applies to orthogonal pairs of Latin squares: To make the larger pair Latin, we can provide a different "offset" for the small squares in any row or column. The pattern of this offset is itself an orthogonal pair of Latin squares, so for offsets we can simply multiply the elements of some pair by the order of the squares, as shown in figure 5.

```
|   The Construction of Orthogonal Offset Values    Fig. 5
|
|   0 1 2 3    0 1 2 3       00 11 22 33          00 44 88 cc
|   1 0 3 2    3 2 1 0   =   13 02 31 20  * 4 =   4c 08 c4 80
|   2 3 0 1    1 0 3 2       21 30 03 12          84 c0 0c 48
|   3 2 1 0    2 3 0 1       32 23 10 01          c8 8c 40 04
```

We can take 16 orthogonal pairs of order 4, and arrange them to produce a larger pair of order 16, using the previously-computed offset values, as shown in figure 6.

```
|   The Orthogonal Checkerboard    Fig. 6
|
| 00+00 11 22 33  44+00 11 22 33  88+00 11 22 33  cc+00 11 22 33
|    13 02 31 20     13 02 31 20     13 02 31 20     13 02 31 20
|    21 30 03 12     21 30 03 12     21 30 03 12     21 30 03 12
|    32 23 10 01     32 23 10 01     32 23 10 01     32 23 10 01
|
| 4c+00 11 22 33  08+00 11 22 33  c4+00 11 22 33  80+00 11 22 33
|    13 02 31 20     13 02 31 20     13 02 31 20     13 02 31 20
|    21 30 03 12     21 30 03 12     21 30 03 12     21 30 03 12
|    32 23 10 01     32 23 10 01     32 23 10 01     32 23 10 01
|
| 84+00 11 22 33  c0+00 11 22 33  0c+00 11 22 33  48+00 11 22 33
|    13 02 31 20     13 02 31 20     13 02 31 20     13 02 31 20
|    21 30 03 12     21 30 03 12     21 30 03 12     21 30 03 12
|    32 23 10 01     32 23 10 01     32 23 10 01     32 23 10 01
|
| c8+00 11 22 33  8c+00 11 22 33  40+00 11 22 33  04+00 11 22 33
|    13 02 31 20     13 02 31 20     13 02 31 20     13 02 31 20
|    21 30 03 12     21 30 03 12     21 30 03 12     21 30 03 12
|    32 23 10 01     32 23 10 01     32 23 10 01     32 23 10 01
```

Now we do the sums in hex and get the results shown in figure 7.

```
|   The Orthogonal Checkerboard Result    Fig. 7
|
|    00 11 22 33     44 55 66 77     88 99 aa bb     cc dd ee ff
|    13 02 31 20     57 46 75 64     9b 8a b9 a8     df ce fd ec
|    21 30 03 12     65 74 47 56     a9 b8 8b 9a     ed fc cf de
|    32 23 10 01     76 67 54 45     ba ab 98 89     fe ef dc cd
|
|    4c 5d 6e 7f     08 19 2a 3b     c4 d5 e6 f7     80 91 a2 b3
|    5f 4e 7d 6c     1b 0a 39 28     d7 c6 f5 e4     93 82 b1 a0
|    6d 7c 4f 5e     29 38 0b 1a     e5 f4 c7 d6     a1 b0 83 92
|    7e 6f 5c 4d     3a 2b 18 09     f6 e7 d4 c5     b2 a3 90 81
|
|    84 95 a6 b7     c0 d1 e2 f3     0c 1d 2e 3f     48 59 6a 7b
|    97 86 b5 a4     d3 c2 f1 e0     1f 0e 3d 2c     5b 4a 79 68
|    a5 b4 87 96     e1 f0 c3 d2     2d 3c 0f 1e     69 78 4b 5a
|    b6 a7 94 85     f2 e3 d0 c1     3e 2f 1c 0d     7a 6b 58 49
|
|    c8 d9 ea fb     8c 9d ae bf     40 51 62 73     04 15 26 37
|    db ca f9 e8     9f 8e bd ac     53 42 71 60     17 06 35 24
|    e9 f8 cb da     ad bc 8f 9e     61 70 43 52     25 34 07 16
|    fa eb d8 c9     be af 9c 8d     72 63 50 41     36 27 14 05
```

The result in figure 7 is a orthogonal pair of Latin squares of order 16 which exhibits massive structure at all levels. In practice we would select each of the 17 constructing pairs at random, and would also permute the rows, columns, and symbols in the resulting order 16 squares. This will produce a reversible discrete combining function which is inherently balanced, and which is just one of a myriad of such functions.

Keyspace

We use order-4 orthogonal Latin squares to build an order-16 Latin square. We do this by selecting from among the 6912 orthogonal pairs for each of 16 positions and the offset square. This is 6912**17 choices which is about 10**65 or 216 bits. Then we shuffle the resulting order-16 square in 16!*15! ways which is about 10**25 or 84 bits. This is a total keyspace of about 300 bits per constructed order-16 orthogonal Latin square. In general we will construct and use an array of such squares.

It should be unnecessary to point out that this keyspace does not represent the "strength" of the nonlinear BBM structure, but instead generally describes the extent to which the resulting 256 bytes of data are unknown. The similar description for an 8-bit substitution table -- with the same amount of data -- is about 1684 bits, and similarly does not represent "strength," unless the table is so well isolated that the precisely correct table can only be found by trial and error.

NONLINEARITY

Boolean Function Nonlinearity

One of the forms of strength in a block cipher is an inability to extrapolate from known parts of the transformation (known plaintext) to model or approximate the transformation at other points (message ciphertexts). Boolean function nonlinearity measures the ability to do such modeling -- for any bit of the result -- based on linear functions. This is a good thing to measure, because the checkerboard construction presumably could have some unnoticed linear structure that could affect the cipher. And nonlinearity measurement can be applied to permutations of reasonable size, including both tables and ciphers.

(Another form of strength is a lack of correlation between key and transformation. Since Mixing ciphers generally use a separate user-key processing and table-shuffling system, the user key itself is well isolated. But correlation between the cipher transformation and table contents is of course an issue in any construction which uses small tables. It is not yet clear how to measure this or any related quantity.)

Specifically, a function is called "Boolean" when it produces just a single result bit. When a permutation is represented in binary, each bit-position can be seen as a different Boolean function. Each of these can be analyzed to find the number of bits which differ between that function and each possible "affine" function. Then the smallest value is the distance in bits to the closest "linear" function. (For an active measurement tool, see:

http://www.io.com/~ritter/JAVASCRP/NONLMEAS.HTM).

We can experimentally measure the extent to which 8-bit ciphering constructs, based on 4-bit tables and mixings, have the same nonlinearity distribution as random 8-bit tables. The experiments will consist of trials of 10,00 constructions each. The reference distribution (taken from the nonlinearity measurements of a single run of 1,000,000 random tables) is shown in figure 8.

```
|  Nonlinearity Distribution of 8-Bit Tables    Fig. 8
|
|  0.35  |                         *
|  0.3   |                         *
|  0.25  |                      *  *
|  0.2   |                      *  *  *
|  0.15  |                      *  *  *
|  0.1   |                   *  *  *  *
|  0.05  |                *  *  *  *  *  *
|  0.00  |  *  *  *  *  *  *  *  *  *  *  *  *
|  Prob  +--+--+--+--+--+--+--+--+--+--+--+--+--
|           84    88    92    96    100   104   Nonlinearity
```

The first component is a keyed or "random" 4-bit invertible substitution table. But 4-bit tables are very, very weak, as shown in figure 9.

```
|  Nonlinearity Distribution of 4-Bit Tables    Fig. 9
|
|   0.6 |
|   0.5 |     *  *
|   0.4 |     *  *
|   0.3 |     *  *
|   0.2 |     *  *
|   0.1 |     *  *
|   0.0 |  *  *  *
|  Prob +--+--+--+--
|          0  2  4   Nonlinearity
```

When we place two 4-bit tables across, we do get the higher values of figure 10, but still only 3 different values. Each of these is exactly 16 times the values from figure 9. This appears to be the effect of repeating a table 16 times, which is what happens with this two-table-wide construction.

```
|  NL Dist. of Two 4-Bit Tables Across 8 Bit Width    Fig. 10
|
|   0.7 |         *
|   0.6 |         *
|   0.5 |         *
|   0.4 |         *
|   0.3 |         *      *
|   0.2 |         *      *
|   0.1 |         *      *
|   0.0 |  *      *      *
|  Prob +--+--//--+--//--+--
|          0      32     64   Nonlinearity
```

The other component is a 4-bit keyed BBM constructed by checkerboard techniques, and producing the nonlinearity distribution of figure 11.

```
|  Nonlinearity of Single Nybble Mixing    Fig. 11
|
|  0.45  |                               *
|  0.40  |                               *
|  0.35  |                               *
|  0.3   |                               *  *
|  0.25  |                               *  *
|  0.2   |                               *  *
|  0.15  |                            *  *  *
|  0.1   |                            *  *  *
|  0.05  |                         *  *  *  *
|  0.00  |  *           *  *  *  *  *  *  *  *  *
|  Prob  +--+--+--+--+--+--+--+--+--+--+--+--+--+--
|              60    68    76    84    92    100    Nonlinearity
```

This is already much nicer than the relatively simple structure from the tables, so one wonders what two such mixings would look like, and the answer is in figure 12.

```
|  Nonlinearity of Two Sequential Nybble Mixings    Fig. 12
|
|  0.35  |                            *
|  0.3   |                            *
|  0.25  |                         *  *
|  0.2   |                         *  *  *
|  0.15  |                      *  *  *  *
|  0.1   |                      *  *  *  *
|  0.05  |                      *  *  *  *
|  0.00  |  *  *  *  *  *  *  *  *  *  *  *  *
|  Prob  +--+--+--+--+--+--+--+--+--+--+--+--+--
|              84    88    92    96    100   104   Nonlinearity
```

Figure 12 is not quite the ideal distribution of figure 8, but it is getting close. And if we put a pair of substitution tables between the mixings, we can get much closer, as shown in figure 13.

```
|  Nonlinearity Distribution of Mix Sub Mix    Fig. 13
|
|  0.35  |                         *
|  0.3   |                         *
|  0.25  |                      *  *
|  0.2   |                      *  *  *
|  0.15  |                      *  *  *
|  0.1   |                   *  *  *  *
|  0.05  |                *  *  *  *  *  *
|  0.00  |  *  *  *  *  *  *  *  *  *  *  *  *
|  Prob  +--+--+--+--+--+--+--+--+--+--+--+--+--
|           84    88    92    96    100   104   Nonlinearity
```

At this point we are beyond what a crude diagram can do, so we enter the realm of numbers, in figure 14.

```
|   Nonlinearity of Mix Sub Mix    Fig. 14
|
|     NL      Expect       Got       ChiSq         Sum  df
|     84:          1         3                }
|     86:          4         3                }
|     88:         13        18       2.000    }  2.000   0
|     90:         46        62       5.565       7.565   1
|     92:        150       187       9.127      16.692   2
|     94:        450       492       3.920      20.612   3
|     96:       1172      1240       3.945      24.557   4
|     98:       2474      2478       0.006      24.564   5
|    100:       3412      3337       1.649      26.212   6
|    102:       2027      1972       1.492      27.705   7
|    104:        249       206       7.426    }
|    106:          2         2       7.367    } 35.071   8
```

This is a typical chi-squared evaluation of "closeness of fit" for this test situation. Normally we expect a chi-squared result of something like the "degrees of freedom" or df, from perhaps half the df to three times the df. Here 35 is high, the distributions differ significantly, and this is typical.

When we add substitutions on the outside of that structure, we do not do much better, as shown in figure 15.

```
|   Nonlinearity of Sub Mix Sub Mix Sub    Fig. 15
|
|     NL      Expect       Got       ChiSq         Sum  df
|     84:          1         2                }
|     86:          4         4                }
|     88:         13        13       0.056    }  0.056   0
|     90:         46        58       3.130       3.186   1
|     92:        150       189      10.140      13.326   2
|     94:        450       471       0.980      14.306   3
|     96:       1172      1235       3.387      17.693   4
|     98:       2474      2592       5.628      23.321   5
|    100:       3412      3325       2.218      25.539   6
|    102:       2027      1896       8.466      34.005   7
|    104:        249       213       5.205    }
|    106:          2         2       5.163    } 39.169   8
```

This is another typical result. But if we add yet another mixing layer and substitution layer, we can get very close indeed, as shown in figure 16.

```
|   Nonlinearity of Sub Mix Sub Mix Sub Mix Sub    Fig. 16
|
|     NL      Expect       Got       ChiSq         Sum  df
|     84:          1         0                 }
|     86:          4         3                 }
|     88:         13        14       0.056     } 0.056   0
|     90:         46        54       1.391       1.447   1
|     92:        150       134       1.707       3.154   2
|     94:        450       483       2.420       5.574   3
|     96:       1172      1185       0.144       5.718   4
|     98:       2474      2436       0.584       6.301   5
|    100:       3412      3436       0.169       6.470   6
|    102:       2027      2004       0.261       6.731   7
|    104:        249       251       0.016     }
|    106:          2         0       0.000     } 6.731   8
```

The chi-squared result from figure 16 has a probability of about 43 percent. This means that, if the two distributions are the same, we should get a chi-squared sum of 6.731 or lower about 43 times out of 100, on average. So we need to run this test a few times.

Figure 17 gives various "critical values" for df = 8, including the percentages that allow us to group results by quarter.

```
|   Chi-Square Percentages for DF = 8    Fig. 17
|
|      1%          5%          95%          99%
|    1.647       2.733      15.507       20.090
|
|          25%         50%         75%
|         5.071       7.344      10.219
```

In figure 18 we group 20 contiguous trials by probability quarter.

```
|   NL ChiSq for Sub Mix Sub Mix Sub Mix Sub; df = 8    Fig. 18
|
|      Q1          Q2          Q3          Q4
|     3.774       5.918       8.083      10.896
|     3.033       5.650       7.876      10.285
|     4.950       5.492       7.691      11.897
|     3.895       5.365
|     4.978       6.634
|     2.978       5.695
|                 6.861
|                 6.746
```

The results in figure 18 seem reasonable: Nothing is suspiciously high, or low. There are more low values than high ones, but we expect variability in these tests of random constructions. The major implication is that the two distributions (the reference distribution taken from random 8-bit tables, and the distribution measured from the constructed 8-bit block cipher) are not significantly different. This is experimental evidence in favor of Mixing cipher construction using nonlinear BBM's.

CONCLUSIONS

With respect to Boolean function nonlinearity, these 8-bit Mixing constructions do not differ significantly from the ideal of random 8-bit tables. So in this sense we have succeeded in simulating an 8-bit table using a Mixing construction based on 4-bit tables and 4-bit BBM components.

These results are entirely consistent with the original experiments using linear BBM's (see

http://www.io.com/~ritter/ARTS/MIXNONLI.HTM).

Those experiments also needed 3 mixings to achieve the reference distribution with 4-bit tables. But further experiments with 5-bit tables and a 10-bit block achieved a good distribution in just 2 mixings. This implied that the need for the additional layers was due to the weakness of the 4-bit tables, and not the Mixing construction itself.

The checkerboard process can efficiently construct keyed, nonlinear, Balanced Block Mixers of substantial size. Because these BBM's retain all the mixing properties of linear BBM's, they can directly replace the linear versions in earlier designs (perhaps with some processing-order changes) for increased strength.

Terry Ritter   ritter@io.com   http://www.io.com/~ritter/
Crypto Glossary 1998-08-27: http://www.io.com/~ritter/GLOSSARY.HTM

Terry Ritter, his current address, and his top page.

Last updated: 1998-09-23

Practical Latin Square Combiners

A *Ciphers By Ritter* Page

Terry Ritter

The strongthru-exclusive-OR combiner often used in stream ciphers can be defined using a Latin square combiner with strength by folding the stream cipher "confusion sequence" or "running key" from a known-plaintext attack into a Latin square combiner. Although the Latin square combiner is random working on byte values can be reduced to 128 bytes by using only the lower half of the Latin square. Further, the Latin square combiner's necessary size can be constructed in a "checkerboard" process of combining two smaller Latin squares with whole Latin squares and adjusting their symbol sets.

INTRODUCTION

Latin Squares

A "Latin square" of "order n" is an n x n array of n distinct symbols, in which each symbol occurs exactly once in each row and column. See Figure 1:

```
  Figure 1.
  Latin Square

  |  1  2  3  4
  |
  |  2  4  1  3
  |
  |  3  1  4  2
  |
  |  4  3  2  1
```

The particular square in Figure 1 is "Latin," in the sense that each row below the top is a rotated version of the row above it. This is not a necessary characteristic of Latin squares, but it is undesirable for cryptography, since the resulting squares are very predictable.

Latin Square Combiners

Confusion and diffusion can be combined in a Latin square: one row selects a row, the other selects a column, and the result is the selected element. If the confusion sequence selects columns, and the plaintext selects rows, the ciphertext is the selected element, as it calculates the inverse of each column in the original square.

A Latin square is an example of "balanced" combining so that every received value occurs with the same probability for any particular output value can be produced by any value on one input, with some value on the other input. This is a kind of "perfect" cryptographic "confusion", but with a large alphabet and the opportunity to have "diffusion" among the symbols. Although the Latin square combiner is a nonlinear exclusive-OR's, I if the combiner has substantial keyed randomness, it should be difficult to cryptanalyze even when the confusion sequence is it would with an exclusive-OR combiner.

Standard Form

In any Latin square, the rows can be re-arranged and the result is still a Latin square. The columns can be re-arranged and the result is still a Latin square. The symbols can be re-labeled and the result is still a Latin square. This allows any Latin square to be put into a "standard form" in which the first row and down the left column, the symbols occur in order across the top row and down the left column. The process of order 4 reduce to the 4 standard squares in Figure 2:

```
  Figure 2
  The Standard Latin Squares of Order 4

  |  1 2 3 4     1 2 3 4     1 2 3 4     1 2 3 4
  |
  |  2 1 4 3     2 4 1 3     2 3 4 1     2 4 1 3
  |
  |  3 4 1 2     3 1 4 2     4 1 2 3     3 1 4 2
  |
  |  4 3 2 1     4 3 2 1     3 4 1 2     4 3 2 1
```

SHUFFLING LATIN SQUARES

A Latin square of order n can be shuffled into n!(n-1)! different squares. For order 4, there are 576 such squares. Although not a unique final, we can perform the remapping column in to-list ways, which is perhaps the simplest. So this will produce n! squares.

We can obtain all possible squares of same order simply shuffling a single Latin square. A Latin square of order 4 can be shuffled into 24 different squares. Although there are 576 such, there are 576 squares in order 4. All 576 squares can be produced by shuffling the 4 different squares of standard form.

The obvious way to shuffle squares is to simulate the structure of a Latin square combiner, and use the combiner's key to drive a usual shuffle process. An alternative is to shuffle tables which represent the Latin square combiner's structure, because then those construct the square indirectly through these transformations.

THE CHECKERBOARD CONSTRUCTION

One way to construct a larger square is to take some Latin square and replace each of the symbols with a full Latin square. By giving each of the symbols in the original square a different set of symbols in the unique in each row and column, and so produce a Latin square of larger size.

If we consider squares with numeric symbols, we can give each symbol a numeric offset to produce a new symbol in the expanded Latin square. We can obtain offset values by multiplying the elements of a square by its order, as in Figure 3:

```
  Figure 3
  The Construction of Offset Values

  |  1 2 3 4     0 0 0 0     1 2 3 4
  |
  |  2 1 4 3     4 4 4 4     6 5 8 7
  |
  |  3 4 1 2     8 12 8 4    11 16 9 6
  |
  |  4 3 2 1     12 8 4 0    16 11 6 1
```

We can use the same original square for all of the replacement squares, as in Figure 4:

```
  Figure 4
  The Checkerboard Construction

  |  0+ 0 1 2 3     4+ 0 1 2 3     8+ 0 1 2 3    12+ 0 1 2 3
  |
  |  0+ 2 3 1 0     4+ 6 5 8 7     8+ 8 9 6 11   12+ 4 5 1 0
  |
  |  0+ 3 2 0 1     4+ 7 8 5 6     8+ 12 6 9 4   12+ 5 1 4 8
  |
  |  0+ 1 0 3 2     4+ 5 6 7 8     8+ 4 11 6 9   12+ 1 0 5 4
```

which produces the order-16 square of Figure 5:

```
  Figure 5
  The Checkerboard Result

  |  0 1 2 3     4 0 1 2     8 0 1 2     12 0 1 2 3
  |
  |  0 2 3 1     6 4 5 7     8 10 9 11   12 4 5 1 0
  |
  |  0 3 2 0     7 4 8 5     8 12 6 9    12 5 1 4 8
  |
  |  0 1 0 3     5 4 6 7     8 4 11 6    12 1 0 5 4

  |  4 5 6 7     8 9 10 11   12 13 14 15    0 1 2 3
  |
  |  4 7 8 5     10 11 9 8   14 15 12 13    2 3 0 1
  |
  |  4 6 7 8     9 11 8 10   14 15 13 12    3 0 1 2
  |
  |  4 5 4 7     8 9 11 10   12 13 15 14    1 2 3 0

  |  8 9 10 11   12 13 14 15    0 1 2 3     4 5 6 7
  |
  |  8 12 6 9    14 15 13 12    2 3 0 1     6 7 4 5
  |
  |  8 11 9 10   15 12 14 13    3 0 1 2     7 4 5 6
  |
  |  8 9 11 6    13 12 15 14    1 2 3 0     5 6 7 4

  |  12 13 14 15    0 1 2 3     4 5 6 7     8 9 10 11
  |
  |  12 4 5 1       2 3 0 1     6 7 4 5     10 11 9 8
  |
  |  12 5 1 4       3 0 1 2     7 4 5 6     11 9 8 10
  |
  |  12 1 0 5       1 2 3 0     5 6 7 4     9 8 11 10
```

Clearly, this Latin square exhibits massive structure at all levels. A Latin square we would use a different order-4 table for each square, and would also shuffle the order-4 tables. We could also shuffle all rows, columns, and symbols in the larger square.

Keyspace

There are 576 Latin squares of order 4, any one of which can be used to produce the order-16 square. The offset square is another 576 Latin squares, and any of these can be used. So there are 576^2=331,776 different squares of order 16 in this way. There up to 24 different squares of order 4 that can be used, so there are 24^2 different ways, thus producing about 2 10^13 squares, for about 44 bits of keyspace. Although this is a very small keyspace, if we only use the 24 order-4 squares formed by shuffling a single standard square, we still have 24! keyspace; we could store 24 order-4 tables in 24 x 16 = 384 bytes. We could also shuffle in an "ALL" combiner, and might even select a combiner dynamically from an array of such tables.

LARGER SQUARE WHOLE CONSTRUCTED SQUARES

We would like to be able to compose the "checkerboard" construction to a "random" construction. Unfortunately, it is not at all clear what a random Latin square construction would be. Various complex ways to construct random Latin squares have been in the literature, but so many squares are possible (there may be 2^100 different Latin squares of order 10) that it is difficult to know whether we really have a random result. We can shuffle a Latin square by randomizing its tables, by defining an appropriate quality and then measuring the quality of the result, or by using a particular well-defined construction. Here we have the "checkerboard" construction and a particular Latin square combiner with a measurable quality to measure might be "Boolean function nonlinearity."

Boolean function Nonlinearity

A function is called "Boolean" when it produces just a single result bit. When a permutation is represented in binary, each bit position in the result can be considered a Boolean function. Latin square is analyzed to find the number of bit which differ between that function and all possible linear functions. The minimum value is the nonlinear bits to the closest "linear" function, and so is one view of "strength." (For an active measurement tool, see:

http://www.ix.com/~ritter/CRYPTMAN/MDA.HTM).

Nonlinearity in Latin Squares

Since each Latin square consists of symbols in some permutation in each row and column, we can apply a Boolean function nonlinearity measure to the rows of a Latin square. Although the rows of Latin squares are very weak, an order 4 square has permutations of length 4 which can be 2 bits from any linear function. While this situation is somewhat better at order 16, these are still very small permutations with a maximum nonlinearity of 4.

When we measure the 16 rows and 16 columns of a square of order 16, it is common for at least one of these to be linear. So the minimum value is zero. The maximum nonlinearity is 32 (one plus 16 columns of 8 bits each), we will occasionally get a result of 32 for a Latin square of order 16. Although higher values of nonlinearity values are always even). This is not much range for use in establishing scarcity in distribution.

In contrast, we collect "both" the minimum value "and" the maximum value, and use the difference between them as the result. For order 16 Latin squares, we have the sum of the nonlinearities for 32 different permutations. This means that the many squares with an overall result of zero cannot be "good" Latin square combiners, but there is a distinct distribution with a reasonable range, and we can use the "checkerboard" construction to produce such a result.

Note that Latin square nonlinearity is not directly comparable to single-permutation nonlinearity, and neither of these is an overall "strength" value: Nonlinearity is just one of the many faces of "strength." Although we would prefer that a Latin square combiner be very nonlinear lower functions and that result be linear, it may turn few and each column will be just such a selection.

Order-16 Latin squares are small (128-byte tables for mixing a byte value), and we can easily store and use tables for mixing bytes directly. But these tables are only part of cipher strength, and we may prefer to use smaller tables in order to provide support dynamic selection from among an array of such tables.

MEASURED LATIN SQUARE NONLINEARITY

The ad hoc construction was used to develop a reference distribution. This involved (only) two tracks of 1,000,000 samples each, with the "good" square and the "bad" square. The "good" square was a square "both" the "checkerboard" and the "ad hoc" experimental distributions to those reference expectations.

Ideally we would find that each checkerboard trial would represent some "good" square, and that the distribution of the "bad" square would in not be the same. Here we have two of the three examples:

In the first bad example (Figure 6), we see over half gaps as many "bad" square nonlinearities as expected of a 1M of our 1M trials (instead of 20, not of 1000), and a similar excess at 64,100 (50 instead of 20):

```
  | Nonlinearity for 1000 Checkerboard Constructions   Fig. 6
  |
  |
  |  No.   Expect    Get     Chi2q        Sum df
  |
  |  70:     0       1     ]
  |
  |  76:     0       1     ]
  |
  |  80:     1       1     ]
  |
  |  84:    12      10     ] 0.800    0.800   0
  |
  |  88:    41      40     ] 0.024    0.824   1
  |
  |  92:    85      91     ] 0.424    1.248   2
  |
  |  96:    95      91     ] 0.016    1.264   3
  |
  |  100:   137     137     ] 0.000    1.264   4
  |
  |  104:   137     115     ] 3.524    4.788   5
  |
  |  108:   132     142     ] 0.758    5.608   6
  |
  |  112:   101     102     ] 0.010    5.618   7
  |
  |  116:   121     132     ] 0.998    6.616   8
  |
  |  120:    78      85     ] 0.312    6.928   9
  |
  |  124:    70      66     ] 0.242    7.170  10
  |
  |  128:    38      40     ] 0.106    7.276  11
  |
  |  132:    12      13     ]
  |
  |  136:     5       4     ]
  |
  |  140:     1       0     ]
  |
  |  144:     1       0     ] 4.050   20.005  13
  |
  |  144:     1       0     ]
```

But in the second bad example (Figure 7), the problem is confined to an No. of 100:

```
  |  Nonlinearity for 1000 Checkerboard Constructions   Fig. 7
  |
  |
  |  No.   Expect    Get     Chi2q        Sum df
  |
  |  80:     1       1     ]
  |
  |  84:     2       2     ]
  |
  |  88:    41      40     ]
  |
  |  92:    12      14     ] 0.800    0.800   0
  |
  |  96:    91      91     ] 0.000    0.800   1
  |
  |  100:   137     137     ] 0.000    0.800   2
  |
  |  104:   115      93     ] 4.356    5.156   3
  |
  |  108:   142     142     ] 0.000    5.156   4
  |
  |  112:   102     102     ] 0.000    5.156   5
  |
  |  116:   132     114     ] 2.455    7.617   6
  |
  |  120:    85      85     ] 0.000    7.617   7
  |
  |  124:    66      70     ] 0.242    7.859   8
  |
  |  128:    40      35     ] 0.692   10.072   9
  |
  |  132:    13      12     ] 0.077   10.149  10
  |
  |  136:     4       5     ] 0.224   10.373  11
  |
  |  140:     0       1     ] 1.000   11.373  12
  |
  |  144:     0       1     ] 0.998   12.371  13
  |
  |  144:     1       0     ]
  |
  |  148:     1       0     ] 0.998   22.667  13
```

In this case both "bad" distributions do have a high count for zero, and both have a high count for maximum nonlinearity. Other than this, there seem to be no clear patterns.

If we collect the chi-square results for 20 tracks of 1000 checkerboard trials, we can see the distribution of results for the two demonstrations problems. In Figure 8 through 10 we see two groups each for both the "ad hoc" and "checkerboard" constructions:

```
  Chi-Square Percentages for DF = 13   Fig. 8

                 20%       40%       60%
  Chi2q     0.250sd   1.0sd     1.5sd

  Checkerboard Chi-Square 10 Trials; df = 13   Fig. 9

      Q1        Q2        Q3        Q4
  3.840    12.953    13.134    16.027
  8.200    13.341    13.195    17.030
  9.488    13.362    15.086    21.026
  9.700    13.395    15.291    21.576
  9.949    13.488    15.507
  9.968    13.498    15.618
  9.515

  Ad Hoc Chi-Square 10 Trials; df = 13   Fig. 10

      Q1        Q2        Q3        Q4
  7.575    13.504    16.775    22.663
  8.165    13.946    19.261
  9.501    13.811    19.931
  9.488    13.908    19.526
  7.412    13.526    19.520
  6.926

  Checkerboard Chi-Square 10 Trials; df = 13   Fig. 11

      Q1        Q2        Q3        Q4
  8.165    12.953    16.027    24.736
  8.600    12.298    13.507    24.312
  9.488    12.      13.195    23.209
  9.110            14.750    21.879
  9.914                      21.000
                             20.090
                             19.655
                             17.462
                             19.576

  Ad Hoc Chi-Square 10 Trials; df = 13   Fig. 12

      Q1        Q2        Q3        Q4
  8.463    11.575    12.985    17.377
  7.488    11.488    13.488    17.488
  7.488    11.488    13.488    17.275
  6.681    11.337    15.301    21.587
  6.507    11.488    15.
                     12.249
```

And if we go to 10,000 constructions per trial, we get the results shown in Figures 13 through 15:

```
  Chi-Square Percentages for DF = 17   Fig. 13

                 20%       40%       60%
  Chi2q     0.250sd   1.0sd     1.5sd

  Checkerboard Chi-Square 10K Trials; df = 17   Fig. 14

      Q1        Q2        Q3        Q4
                                  47.628
                                  90.971
                                  52.000
                                  52.000
                                  48.274

  Ad Hoc Chi-Square 10K Trials; df = 17   Fig. 15

      Q1        Q2        Q3        Q4
  19.357    13.944    18.939    28.507
  18.000    13.000    18.000    27.000
  17.000    12.305    18.750    25.653
  17.170    13.545    19.016    27.363
            13.000    18.000    26.000
                      19.115
```

A detailed examination of the outrageous "checkerboard" results in Figure 14 shows that some of these have excessive low-Q counts, others have excessive high-Q counts, while some have variations between these extremes. The "ad hoc" results are more consistent. Therefore, the "bad" square construction, for "checkerboard" results, which is nice, seeing as the reference distribution was developed from that construction.

COMMENTS

These two Latin square constructions produce measurably different distributions. So if we are given a large number of squares, by measuring the distribution of nonlinearity results, we can determine which construction produced those squares. But this is information which is not normally hidden in cryptography anyway.

On the other hand, both constructions deliver a comparable range of nonlinearity results, so the nonlinearity measure value are good. This would seem to be evidence supporting both constructions for cryptographic use. The "checkerboard" construction is easier to comparatively easy and efficient way to produce nonlinear Latin square combiners of useful size.

Terry Ritter ritter@ix.com http://www.ix.com/~ritter/
Crypto Glossary 1994-06-27 http://www.ix.com/~ritter/BLOGSARY.HTM

Terry Ritter ritter@ix.com http://www.ix.com/~ritter/
Crypto Glossary 1994-06-27 http://www.ix.com/~ritter/BLOGSARY.HTM

From: "Pierre Abbat"
Subject: Re: Practical Latin Square Combiners
Newsgroups: sci.crypt
Message-ID: <3D7F75ca.2575040@news.ix.netcom> <3D7F75ca.2575040@news.ix.com>
References: <6t1ka2ca1ka4ka4ka4ka4ka@news.ix.netcom.com>
Date: Fri, 10 Sep 1993 01:42:03 GMT

```
|  The Standard Latin Square of Order 4   Fig. 1
|
|  1 2 3 4     1 2 3 4     1 2 3 4     1 2 3 4
|
|  2 1 4 3     2 4 1 3     2 3 4 1     2 4 1 3
|
|  3 4 1 2     3 1 4 2     4 1 2 3     3 1 4 2
|
|  4 3 2 1     4 3 2 1     3 4 1 2     4 3 2 1
```

Correct:

```
|  The Standard Latin Square of Order 4   Fig. 2
|
|  1 2 3 4     1 2 3 4     1 2 3 4     1 2 3 4
|
|  2 1 4 3     2 4 1 3     2 3 4 1     2 4 1 3
|
|  3 4 1 2     3 1 4 2     4 1 2 3     3 1 4 2
|
|  4 3 2 1     4 3 2 1     3 4 1 2     4 3 2 1
```

Terry Ritter ritter@ix.com http://www.ix.com/~ritter/
Crypto Glossary 1994-06-27 http://www.ix.com/~ritter/BLOGSARY.HTM

From: "Terry Ritter"
Subject: Re: Practical Latin Square Combiners
Newsgroups: sci.crypt
Message-ID: <3D7F75ca.2575040@news.ix.netcom>
References: <3D7F75ca.2575040@news.ix.com>
Date: Wed, 10 Sep 1993 03:20:00 GMT, on <3D7F75ca.1525870@news.ix.com>,
in <6t1ka2ca.2ka4ka4ka4ka4ka@news.ix.netcom.com>

>PRACTICAL LATIN SQUARE COMBINERS
>[...]

The 3rd square in Figure 2 had a type:

Original:

```
|  The Standard Latin Square of Order 4   Fig. 2
|
|  1 2 3 4     1 2 3 4     1 2 3 4     1 2 3 4
|
|  2 1 4 3     2 4 1 3     2 3 4 1     2 4 1 3
|
|  3 4 1 2     3 1 4 2     4 1 2 3     3 1 4 2
|
|  4 3 2 1     4 3 2 1     3 4 1 2     4 3 2 1
```

Correct:

```
|  The Standard Latin Square of Order 4   Fig. 2
|
|  1 2 3 4     1 2 3 4     1 2 3 4     1 2 3 4
|
|  2 1 4 3     2 4 1 3     2 3 4 1     2 4 1 3
|
|  3 4 1 2     3 1 4 2     4 1 2 3     3 1 4 2
|
|  4 3 2 1     4 3 2 1     3 4 1 2     4 3 2 1
```

Terry Ritter ritter@ix.com http://www.ix.com/~ritter/
Crypto Glossary 1994-06-27 http://www.ix.com/~ritter/BLOGSARY.HTM

The 3rd square in Figure 2 has a type:

```
|  The Standard Latin Square of Order 4   Fig. 2
|
|  1 2 3 4     1 2 3 4     1 2 3 4     1 2 3 4
|
|  2 1 4 3     2 4 1 3     2 3 4 1     2 4 1 3
|
|  3 4 1 2     3 1 4 2     4 1 2 3     3 1 4 2
|
|  4 3 2 1     4 3 2 1     3 4 1 2     4 3 2 1
```

Original:

```
|  The Standard Latin Square of Order 4   Fig. 2
|
|  1 2 3 4     1 2 3 4     1 2 3 4     1 2 3 4
|
|  2 1 4 3     2 4 1 3     2 3 4 1     2 4 1 3
|
|  3 4 1 2     3 1 4 2     4 1 2 3     3 1 4 2
|
|  4 3 2 1     4 3 2 1     3 4 1 2     4 3 2 1
```

Correct:

```
|  The Standard Latin Square of Order 4   Fig. 2
|
|  1 2 3 4     1 2 3 4     1 2 3 4     1 2 3 4
|
|  2 1 4 3     2 4 1 3     2 3 4 1     2 4 1 3
|
|  3 4 1 2     3 1 4 2     4 1 2 3     3 1 4 2
|
|  4 3 2 1     4 3 2 1     3 4 1 2     4 3 2 1
```

Terry Ritter ritter@ix.com http://www.ix.com/~ritter/
Crypto Glossary 1994-06-27 http://www.ix.com/~ritter/BLOGSARY.HTM

Three other ways to transform one Latin square into another are to invert the permutation in each row, to permute the numbers, and to transpose it. Under these operations the first three squares are equivalent:

```
1 2 3 4
2 1 4 3
3 4 1 2
4 3 2 1
```

by exchanging 3 and 2 becomes

```
1 3 2 4
2 1 4 3
3 4 1 2
4 2 3 1
```

which by exchanging rows and exchanging columns becomes

```
1 2 3 4
2 4 1 3
3 1 4 2
4 3 2 1
```


Copyright © 1994 Terry Ritter. All rights reserved.

Break This 8-Bit Block Mixing Cipher

A *Ciphers By Ritter* Page

Terry Ritter

Contents

- Block Cipher Design Page: Find the Wide Block Cipher, its original series
- Cipher Mailing Mailing Page: To add me to a mailing or the cipher-list you prefer, give your email or the phrase (cipherlist-yes, cl-fast-yes, phrase-cipherlist-yes,) and
- Cipher Mailing Mailing Page: To remove your email from the mailing, give your email or the phrase (cipherlist-no, cl-fast-no, phrase-cipherlist-no,)
- Cipher Mailing Mailing Page: To see the current mailing lists, give the phrase (cipherlist-show, cl-fast-show, phrase-cipherlist-show,)

Abstract

A 8-bit block "Mixing" block cipher structure has "exchanged" all the bits in its block every round. This structure is quite simple, and has been proposed as a "fast" cipher for 8-bit processors. Although the structure has been known for years, it has never been analyzed in a reasonable manner. This page shows a simple attack on the structure, and suggests that the structure may be appropriate for the study of cryptanalytic principles, since it is a simple structure which is not easy to break.

The 8-bit block is one of the smallest practical block sizes. A "substitution" cipher with 8-bit blocks is a simple table-lookup, but such a cipher is not cryptanalysis when perfect security is known. Unfortunately, "weak" substitution ciphers are not cryptanalysis when they are easily broken, but such ciphers are not cryptanalysis because there is no "cryptanalytic" method which can be used to demonstrate the weakness of such a cipher. Thus, we have a problem in that we can not study cryptanalytic principles in the 8-bit block case.

A previously proposed proposed a cipher in 8-bit block sizes which is quite simple and which has been proposed for use in 8-bit processors. This cipher has been known for years, but has never been analyzed in a reasonable manner. This design, since known attacks are known or expected to this 8-bit design, is a good candidate for study of cryptanalysis.

The Problem

We have a block cipher composed of four unrelated substitution-tables, and a simple "mixing" structure which "exchanges" all the bits in the block every round. This structure is called "Mixing" and is described in the Block Cipher Design page:

http://www.ciphers.com/cipher/BlockCipher.htm

It is known that we have quite practical attacks against substitution ciphers with substitution tables which are fixed. This page shows that the "mixing" structure does not add any strength to the cipher, and that the cipher is easily broken. This page shows that the "mixing" structure does not add any strength to the cipher, and that the cipher is easily broken.

The 8-bit block is a reasonable block size for a cipher which is easy enough to analyze that practical attacks are known, but which is not so small that it is easily broken.

Cryptanalysis

The study of "cryptanalysis" is this which is "not" the protection of information against the attacks of "weak" or "broken" ciphers. The substitution of an 8-bit structure, and so on both the substitution and the "mixing" structure, is a "weak" or "broken" cipher, and is not cryptanalysis because there is no "cryptanalytic" method which can be used to demonstrate the weakness of such a cipher.

"Part" of the meaning of "cryptanalysis" is "weak ciphers". The 8-bit block is a "weak" cipher because it is easily broken, and because there is no "cryptanalytic" method which can be used to demonstrate the weakness and the true strength in an 8-bit block cipher.

The "Mixing" structure is a "weak" cipher because it is easily broken. A simple form of substitution structure which is "weak" or "broken" is a "substitution" cipher. A "substitution" cipher is a simple table-lookup which accepts a strength value of such the 8-bit block for each cipher. The "substitution" cipher is a "weak" or "broken" cipher because it is easily broken. A "substitution" cipher, "is" a "weak" or "broken" cipher because it is easily broken. The "mixing" structure is a "weak" or "broken" cipher because it is easily broken. The "mixing" structure is a "weak" or "broken" cipher because it is easily broken.

The meaning of "cryptanalysis" is this which "is" the protection of information against the attacks of "weak" or "broken" ciphers. The substitution of an 8-bit structure, and so on both the substitution and the "mixing" structure, is a "weak" or "broken" cipher, and is not cryptanalysis because there is no "cryptanalytic" method which can be used to demonstrate the weakness of such a cipher.

"Part" of the meaning of "cryptanalysis" is "weak ciphers". The 8-bit block is a "weak" cipher because it is easily broken, and because there is no "cryptanalytic" method which can be used to demonstrate the weakness and the true strength in an 8-bit block cipher.

The "Mixing" structure is a "weak" cipher because it is easily broken. A simple form of substitution structure which is "weak" or "broken" is a "substitution" cipher. A "substitution" cipher is a simple table-lookup which accepts a strength value of such the 8-bit block for each cipher. The "substitution" cipher is a "weak" or "broken" cipher because it is easily broken. A "substitution" cipher, "is" a "weak" or "broken" cipher because it is easily broken. The "mixing" structure is a "weak" or "broken" cipher because it is easily broken. The "mixing" structure is a "weak" or "broken" cipher because it is easily broken.

The meaning of "cryptanalysis" is this which "is" the protection of information against the attacks of "weak" or "broken" ciphers. The substitution of an 8-bit structure, and so on both the substitution and the "mixing" structure, is a "weak" or "broken" cipher, and is not cryptanalysis because there is no "cryptanalytic" method which can be used to demonstrate the weakness of such a cipher.

"Part" of the meaning of "cryptanalysis" is "weak ciphers". The 8-bit block is a "weak" cipher because it is easily broken, and because there is no "cryptanalytic" method which can be used to demonstrate the weakness and the true strength in an 8-bit block cipher.

The "Mixing" structure is a "weak" cipher because it is easily broken. A simple form of substitution structure which is "weak" or "broken" is a "substitution" cipher. A "substitution" cipher is a simple table-lookup which accepts a strength value of such the 8-bit block for each cipher. The "substitution" cipher is a "weak" or "broken" cipher because it is easily broken. A "substitution" cipher, "is" a "weak" or "broken" cipher because it is easily broken. The "mixing" structure is a "weak" or "broken" cipher because it is easily broken. The "mixing" structure is a "weak" or "broken" cipher because it is easily broken.

Scalability

Scalability is a historical new concept in cipher design, so it is not well-understood. Nevertheless, it is important to know that a cipher design is a simple cipher design which be proven a real-size design. Thus, we need to know that a cipher design is a simple cipher design which be proven a real-size design. It is desirable to know that a cipher design is a simple cipher design.

The point of a scally-scalable design is to have a fixed definition which can be proven to be a real-size design. Unfortunately, this is not the case with the 8-bit block. Thus, "scalability" is not a real-size design. Thus, "scalability" is not a real-size design.

This particular design consists of exactly two components: random-table lookup, and a "mixing" structure which "exchanges" all the bits. Both cipher and substitution-table are "fixed" and "known" in advance. Thus, there is no "key" in this cipher.

The block is defined differently-to-select to 8-bit processors, so it is not a "real" cipher. Thus, we have a "real" cipher which is not a real-size design. Thus, we have a "real" cipher which is not a real-size design.

The "mixing" structure is a "fixed" and "known" in advance cipher. Thus, there is no "key" in this cipher. Thus, there is no "key" in this cipher. Thus, there is no "key" in this cipher.

If there is a dependence on the larger variation, the size "used" is a "real" cipher. Thus, there is no "key" in this cipher. Thus, there is no "key" in this cipher.

The Design

The block cipher design which has 8-bit tables and data-parts, and which has a "mixing" structure which "exchanges" all the bits is not useful for timing data, but is useful for analysis.

```
                    INPUT BLOCK
                        |
                        |
                        |
```

```
/* Implementation for an Algebraic cipher. The keying tables are set
   up elsewhere.
*/
public Algebraic(
    int[] pTable, int[] pInvTable, int[] kTable,
    int[] kInvTable, int[] mTable, int[] mInvTable, int[] c1,
    int[] c2, int[] c3, int[] c4, int[] c5, int[] c6,
    int[] c7, int[] c8, int[] c9, int[] c10, int[] c11,
    int[] c12, int[] c13, int[] c14, int[] c15, int[] c16
)
{
    /*
    Implementation of private functions.
    */

    Random Random = new Random();

    private ICryptoTransform[] ctr = null;
    private ICryptoTransform[] ctrInv = null;
    private byte[] iv = null;

    public byte[] encrypt(byte[] plaintext, int[] pTable,
        int[] kTable, int[] mTable, int[] c1,
        int[] c2, int[] c3, int[] c4, int[] c5,
        int[] c6, int[] c7, int[] c8, int[] c9,
        int[] c10, int[] c11, int[] c12, int[] c13,
        int[] c14, int[] c15, int[] c16)
    {
        int[] ct = null;
        int[] ct2 = null;
        int[] ct3 = null;
        int[] ct4 = null;
        int[] ct5 = null;
        int[] ct6 = null;
        int[] ct7 = null;
        int[] ct8 = null;
        int[] ct9 = null;
        int[] ct10 = null;
        int[] ct11 = null;
        int[]
```

Break This 4-Bit Block Mixing Cipher

A *Ciphers By Ritter* Page

Terry Ritter

A 4-bit-wide block cipher model is presented. Given all 16 possible messages and their ciphertexts, the goal is to develop the table arrangements which are the cipher key. The tiny size is intended to support a deep understanding of strength, something which to date has been impossible in any full-size cipher. Related 6-bit and 8-bit versions are also mentioned, should the 4-bit version be in some way a special case.

From: ritter@io.com (Terry Ritter)
Newsgroups: sci.crypt
Subject: Break This 4-Bit Block Cipher!
Date: Mon, 30 Mar 1998 04:02:40 GMT
Message-ID: <351f18af.418844@news.io.com>

BREAK THIS 4-BIT BLOCK CIPHER!

Terry Ritter ritter@io.com
Ritter Software Engineering
http://www.io.com/~ritter/

1998-03-29

ABSTRACT

A 4-bit-wide block cipher model is presented. Given all 16 possible messages and their ciphertexts, the goal is to develop the table arrangements which are the cipher key. The tiny size is intended to support a deep understanding of strength, something which to date has been impossible in any full-size cipher. Related 6-bit and 8-bit versions are also mentioned, should the 4-bit version be in some way a special case.

THE PROBLEM

We have a 4-bit block cipher. It is composed of four keyed 2-bit invertible substitution tables (4 entries of 2 bits each), and a linear Balanced Block Mixer. That's it. This is an extremely reduced *model* specifically intended for analysis.

We assume that we know every possible plaintext and ciphertext pair: this is the ultimate known-plaintext and defined-plaintext attack. The goal is to develop the table permutations -- the key -- with some sort of scalable algorithm.

```
|                      INPUT BLOCK
|             InA |                  | InB
|                 v                  v
|             --------             --------
|            |   TA   |          |   TB   |
|             --------             --------
|                 |                  |
|               +--*----------------+  |
|               |                  |  |
|               |    +-----------------*-+
|            A |    | B          A |    | B
|               v    v              v    v
|             --------             --------
|            |   L0   |    BBM    |   L1   |
|             --------             --------
|               X |                  | Y
|                 v                  v
|             --------             --------
|            |   TX   |          |   TY   |
|             --------             --------
|                 |                  |
|           OutX  v                  v  OutY
|                      OUTPUT BLOCK
```

(In the figure, the tables are TA, TB, TX and TY, and the Balanced Block Mixer is shown as two Latin square combiners L0 and L1.)

A STARTING ANALYSIS

Since InA and InB are known, we can write equations for unknown table entries TA[InA] and TB[InB] in terms of known input values:

```
   A = TA[ InA ]
   B = TB[ InB ]
```

On the other hand, although the output values are also known, it might seem that we have to define those values in terms of unknown table positions, as well as unknown values:

```
   OutX = TX[ X ]
   OutY = TY[ Y ]
```

But since we know that the tables are invertible, without loss of generality we can introduce inverse tables IX and IY and write:

```
   X = IX[ OutX ]
   Y = IY[ OutY ]
```

(If we succeed in defining the inverse tables, we can easily develop their inverses, which are the values we want.)

So now we have 4 known external values which select 4 unknown values which have a linear BBM relationship to each other. The orthogonal Latin squares in the BBM are developed algorithmically as follows:

```
   X = 3A + 2B (mod 2)(mod p),  p = 111
   Y = 2A + 3B (mod 2)(mod p)
```

The BBM equations may seem unusual, and this can make it awkward to think about the system. So, instead of the equations, we can use a table. Here A selects a row, B selects a column, and the selected entry is XY. Remember, there are only 4 different values in each 2-bit half-block.

```
         0   1   2   3

   0    00  23  31  12
   1    32  11  03  20
   2    13  30  22  01
   3    21  02  10  33
```

Suppose we concentrate on the case where A = 0: This is the top row of the table. Note that both X and Y take on each of their 4 possible values exactly once. But this happens in *every* row, so any of these give the same results, as long as we do not know the TA keying.

Suppose we hope to identify 00 by the fact that X = Y: But if we look at the table, we find that this *also* happens exactly 4 times, once for each possible value. Again, a suitable keying of TA and TB can produce this same effect for any input.

These simple tests appear to have gotten nowhere. Can other tests improve upon the situation? Can we *prove* that this simple system is, or is not, solvable?

AVOIDING BRUTE FORCE

Since each 2-bit table has 4 elements (with 8 unknown bits), there are 4! or 24 different tables (for about 4.6 bits of keying per selection). We know all 16 of the 4-bit "messages" for a known information total of 64 bits, against the unknown apparent keyspace of about 18 bits. This is brute-force territory, but the intent here is to find an attack which will scale up with the model.

If we are simply unable to resist the pull of brute-force, we should instead consider the similar 6-bit block cipher which uses 3-bit tables (at about 15.3 bits each). Here we know all 64 6-bit messages for a known total of 384 bits, against an unknown apparent keyspace of about 61 bits.

Or we might consider the 8-bit block cipher which uses 4-bit tables (about 44 bits each), and has 256 8-bit messages for a known information total of 2048 bits, against an unknown apparent keyspace of about 177 bits.

Under the given conditions, there is always substantially more known information than the internal state we are trying to define.

COMMENTS

The model is a single linear mixing with unknown tables on both the input and output. This seems to be the ultimate simplification of both the Mixing and VSBC ciphers, and in fact may be a step too far. The real ciphers have at least three layers of tables and two mixings, so that "known plaintext" is not available across any single mixing.

This tiny model demonstrates the analytic advantage of true cipher scalability. One would think that a cipher with a 16-element codebook would be small enough to either break by hand, or know why not. This last possibility would be An Important Result. Presumably, such an outcome would for the first time make it possible to prove strength in a practical full-size cipher, something I never expected to see.

It seems to me that the guaranteed balance of the BBM protects the tables from being separated and exposed. If so, the simple linear BBM contributes to strength in an essential way, despite having absolutely no strength of its own.

Many articles on the larger systems and their general design principles can be found on my web pages. But

```
   http://www.io.com/~ritter/JAVASCRP/ACTIVBBM.HTM
```

has an new introduction to BBM computation, and implements a couple of functional model-size BBM's in JavaScript.

Terry Ritter ritter@io.com http://www.io.com/~ritter/
The Crypto Glossary: http://www.io.com/~ritter/GLOSSARY.HTM

Terry Ritter, his current address, and his top page.

Last updated: 1998-03-31

Measured Distant Avalanche in Large Block Ciphers

A *Ciphers By Ritter* Page

Terry Ritter

Some block ciphers handle data blocks of dramatically variable size. These are not stream ciphers, but are block ciphers with variable block sizes. Such ciphers may offer different avalanche and avalanche behavior for different block sizes. This paper reviews the literature on avalanche in large block ciphers, which suggests the possibility that mixing could be more effective for such ciphers than for ciphers with small block sizes. This paper also reviews the literature on measuring avalanche, and presents a new method for measuring avalanche in large block ciphers. This method is then used to measure avalanche in Dynamic Substitution Combiners (combiners using Dynamic Substitution) and in Dynamic Substitution Ciphers (ciphers using Dynamic Substitution for mixing). Some of these ciphers show very good avalanche behavior, which suggests that Dynamic Substitution may be a very effective mixing method for large block ciphers.

Contents

Text Ciphers: Measured Distant Avalanche in Large Block Ciphers, the original paper
Text Ciphers: Measured Distant Avalanche in Large Block Ciphers, the revised paper
Text Ciphers: What's not a distant problem in block mixing, this video covers the appropriate case
Text Ciphers: Let's Mix Small Blocks!
Text Ciphers: Let's Mix Small Blocks! Part 2
Text Ciphers: Let's Mix Small Blocks! Part 3
Text Ciphers: Let's Mix Small Blocks! Part 4
Text Ciphers: Let's Mix Small Blocks! Part 5
Text Ciphers: Let's Mix Small Blocks! Part 6
Text Ciphers: Let's Mix Small Blocks! Part 7
Text Ciphers: Let's Mix Small Blocks! Part 8
Text Ciphers: Let's Mix Small Blocks! Part 9
Text Ciphers: Let's Mix Small Blocks! Part 10
Text Ciphers: Let's Mix Small Blocks! Part 11
Text Ciphers: Let's Mix Small Blocks! Part 12
Text Ciphers: Let's Mix Small Blocks! Part 13
Text Ciphers: Let's Mix Small Blocks! Part 14
Text Ciphers: Let's Mix Small Blocks! Part 15
Text Ciphers: Let's Mix Small Blocks! Part 16
Text Ciphers: Let's Mix Small Blocks! Part 17
Text Ciphers: Let's Mix Small Blocks! Part 18
Text Ciphers: Let's Mix Small Blocks! Part 19
Text Ciphers: Let's Mix Small Blocks! Part 20
Text Ciphers: Let's Mix Small Blocks! Part 21
Text Ciphers: Let's Mix Small Blocks! Part 22
Text Ciphers: Let's Mix Small Blocks! Part 23
Text Ciphers: Let's Mix Small Blocks! Part 24
Text Ciphers: Let's Mix Small Blocks! Part 25
Text Ciphers: Let's Mix Small Blocks! Part 26
Text Ciphers: Let's Mix Small Blocks! Part 27
Text Ciphers: Let's Mix Small Blocks! Part 28
Text Ciphers: Let's Mix Small Blocks! Part 29
Text Ciphers: Let's Mix Small Blocks! Part 30
Text Ciphers: Let's Mix Small Blocks! Part 31
Text Ciphers: Let's Mix Small Blocks! Part 32
Text Ciphers: Let's Mix Small Blocks! Part 33
Text Ciphers: Let's Mix Small Blocks! Part 34
Text Ciphers: Let's Mix Small Blocks! Part 35
Text Ciphers: Let's Mix Small Blocks! Part 36
Text Ciphers: Let's Mix Small Blocks! Part 37
Text Ciphers: Let's Mix Small Blocks! Part 38
Text Ciphers: Let's Mix Small Blocks! Part 39
Text Ciphers: Let's Mix Small Blocks! Part 40
Text Ciphers: Let's Mix Small Blocks! Part 41
Text Ciphers: Let's Mix Small Blocks! Part 42
Text Ciphers: Let's Mix Small Blocks! Part 43
Text Ciphers: Let's Mix Small Blocks! Part 44
Text Ciphers: Let's Mix Small Blocks! Part 45
Text Ciphers: Let's Mix Small Blocks! Part 46
Text Ciphers: Let's Mix Small Blocks! Part 47
Text Ciphers: Let's Mix Small Blocks! Part 48
Text Ciphers: Let's Mix Small Blocks! Part 49
Text Ciphers: Let's Mix Small Blocks! Part 50
Text Ciphers: Let's Mix Small Blocks! Part 51
Text Ciphers: Let's Mix Small Blocks! Part 52
Text Ciphers: Let's Mix Small Blocks! Part 53
Text Ciphers: Let's Mix Small Blocks! Part 54
Text Ciphers: Let's Mix Small Blocks! Part 55
Text Ciphers: Let's Mix Small Blocks! Part 56
Text Ciphers: Let's Mix Small Blocks! Part 57
Text Ciphers: Let's Mix Small Blocks! Part 58
Text Ciphers: Let's Mix Small Blocks! Part 59
Text Ciphers: Let's Mix Small Blocks! Part 60
Text Ciphers: Let's Mix Small Blocks! Part 61
Text Ciphers: Let's Mix Small Blocks! Part 62
Text Ciphers: Let's Mix Small Blocks! Part 63
Text Ciphers: Let's Mix Small Blocks! Part 64
Text Ciphers: Let's Mix Small Blocks! Part 65
Text Ciphers: Let's Mix Small Blocks! Part 66
Text Ciphers: Let's Mix Small Blocks! Part 67
Text Ciphers: Let's Mix Small Blocks! Part 68
Text Ciphers: Let's Mix Small Blocks! Part 69
Text Ciphers: Let's Mix Small Blocks! Part 70
Text Ciphers: Let's Mix Small Blocks! Part 71
Text Ciphers: Let's Mix Small Blocks! Part 72
Text Ciphers: Let's Mix Small Blocks! Part 73
Text Ciphers: Let's Mix Small Blocks! Part 74
Text Ciphers: Let's Mix Small Blocks! Part 75
Text Ciphers: Let's Mix Small Blocks! Part 76
Text Ciphers: Let's Mix Small Blocks! Part 77
Text Ciphers: Let's Mix Small Blocks! Part 78
Text Ciphers: Let's Mix Small Blocks! Part 79
Text Ciphers: Let's Mix Small Blocks! Part 80
Text Ciphers: Let's Mix Small Blocks! Part 81
Text Ciphers: Let's Mix Small Blocks! Part 82
Text Ciphers: Let's Mix Small Blocks! Part 83
Text Ciphers: Let's Mix Small Blocks! Part 84
Text Ciphers: Let's Mix Small Blocks! Part 85
Text Ciphers: Let's Mix Small Blocks! Part 86
Text Ciphers: Let's Mix Small Blocks! Part 87
Text Ciphers: Let's Mix Small Blocks! Part 88
Text Ciphers: Let's Mix Small Blocks! Part 89
Text Ciphers: Let's Mix Small Blocks! Part 90
Text Ciphers: Let's Mix Small Blocks! Part 91
Text Ciphers: Let's Mix Small Blocks! Part 92
Text Ciphers: Let's Mix Small Blocks! Part 93
Text Ciphers: Let's Mix Small Blocks! Part 94
Text Ciphers: Let's Mix Small Blocks! Part 95
Text Ciphers: Let's Mix Small Blocks! Part 96
Text Ciphers: Let's Mix Small Blocks! Part 97
Text Ciphers: Let's Mix Small Blocks! Part 98
Text Ciphers: Let's Mix Small Blocks! Part 99
Text Ciphers: Let's Mix Small Blocks! Part 100
Text Ciphers: Let's Mix Small Blocks! Part 101
Text Ciphers: Let's Mix Small Blocks! Part 102
Text Ciphers: Let's Mix Small Blocks! Part 103
Text Ciphers: Let's Mix Small Blocks! Part 104
Text Ciphers: Let's Mix Small Blocks! Part 105
Text Ciphers: Let's Mix Small Blocks! Part 106
Text Ciphers: Let's Mix Small Blocks! Part 107
Text Ciphers: Let's Mix Small Blocks! Part 108
Text Ciphers: Let's Mix Small Blocks! Part 109
Text Ciphers: Let's Mix Small Blocks! Part 110
Text Ciphers: Let's Mix Small Blocks! Part 111
Text Ciphers: Let's Mix Small Blocks! Part 112
Text Ciphers: Let's Mix Small Blocks! Part 113
Text Ciphers: Let's Mix Small Blocks! Part 114
Text Ciphers: Let's Mix Small Blocks! Part 115
Text Ciphers: Let's Mix Small Blocks! Part 116
Text Ciphers: Let's Mix Small Blocks! Part 117
Text Ciphers: Let's Mix Small Blocks! Part 118
Text Ciphers: Let's Mix Small Blocks! Part 119
Text Ciphers: Let's Mix Small Blocks! Part 120
Text Ciphers: Let's Mix Small Blocks! Part 121
Text Ciphers: Let's Mix Small Blocks! Part 122
Text Ciphers: Let's Mix Small Blocks! Part 123
Text Ciphers: Let's Mix Small Blocks! Part 124
Text Ciphers: Let's Mix Small Blocks! Part 125
Text Ciphers: Let's Mix Small Blocks! Part 126
Text Ciphers: Let's Mix Small Blocks! Part 127
Text Ciphers: Let's Mix Small Blocks! Part 128
Text Ciphers: Let's Mix Small Blocks! Part 129
Text Ciphers: Let's Mix Small Blocks! Part 130
Text Ciphers: Let's Mix Small Blocks! Part 131
Text Ciphers: Let's Mix Small Blocks! Part 132
Text Ciphers: Let's Mix Small Blocks! Part 133
Text Ciphers: Let's Mix Small Blocks! Part 134
Text Ciphers: Let's Mix Small Blocks! Part 135
Text Ciphers: Let's Mix Small Blocks! Part 136
Text Ciphers: Let's Mix Small Blocks! Part 137
Text Ciphers: Let's Mix Small Blocks! Part 138
Text Ciphers: Let's Mix Small Blocks! Part 139
Text Ciphers: Let's Mix Small Blocks! Part 140
Text Ciphers: Let's Mix Small Blocks! Part 141
Text Ciphers: Let's Mix Small Blocks! Part 142
Text Ciphers: Let's Mix Small Blocks! Part 143
Text Ciphers: Let's Mix Small Blocks! Part 144
Text Ciphers: Let's Mix Small Blocks! Part 145
Text Ciphers: Let's Mix Small Blocks! Part 146
Text Ciphers: Let's Mix Small Blocks! Part 147
Text Ciphers: Let's Mix Small Blocks! Part 148
Text Ciphers: Let's Mix Small Blocks! Part 149
Text Ciphers: Let's Mix Small Blocks! Part 150
Text Ciphers: Let's Mix Small Blocks! Part 151
Text Ciphers: Let's Mix Small Blocks! Part 152
Text Ciphers: Let's Mix Small Blocks! Part 153
Text Ciphers: Let's Mix Small Blocks! Part 154
Text Ciphers: Let's Mix Small Blocks! Part 155
Text Ciphers: Let's Mix Small Blocks! Part 156
Text Ciphers: Let's Mix Small Blocks! Part 157
Text Ciphers: Let's Mix Small Blocks! Part 158
Text Ciphers: Let's Mix Small Blocks! Part 159
Text Ciphers: Let's Mix Small Blocks! Part 160
Text Ciphers: Let's Mix Small Blocks! Part 161
Text Ciphers: Let's Mix Small Blocks! Part 162
Text Ciphers: Let's Mix Small Blocks! Part 163
Text Ciphers: Let's Mix Small Blocks! Part 164
Text Ciphers: Let's Mix Small Blocks! Part 165
Text Ciphers: Let's Mix Small Blocks! Part 166
Text Ciphers: Let's Mix Small Blocks! Part 167
Text Ciphers: Let's Mix Small Blocks! Part 168
Text Ciphers: Let's Mix Small Blocks! Part 169
Text Ciphers: Let's Mix Small Blocks! Part 170
Text Ciphers: Let's Mix Small Blocks! Part 171
Text Ciphers: Let's Mix Small Blocks! Part 172
Text Ciphers: Let's Mix Small Blocks! Part 173
Text Ciphers: Let's Mix Small Blocks! Part 174
Text Ciphers: Let's Mix Small Blocks! Part 175
Text Ciphers: Let's Mix Small Blocks! Part 176
Text Ciphers: Let's Mix Small Blocks! Part 177
Text Ciphers: Let's Mix Small Blocks! Part 178
Text Ciphers: Let's Mix Small Blocks! Part 179
Text Ciphers: Let's Mix Small Blocks! Part 180
Text Ciphers: Let's Mix Small Blocks! Part 181
Text Ciphers: Let's Mix Small Blocks! Part 182
Text Ciphers: Let's Mix Small Blocks! Part 183
Text Ciphers: Let's Mix Small Blocks! Part 184
Text Ciphers: Let's Mix Small Blocks! Part 185
Text Ciphers: Let's Mix Small Blocks! Part 186
Text Ciphers: Let's Mix Small Blocks! Part 187
Text Ciphers: Let's Mix Small Blocks! Part 188
Text Ciphers: Let's Mix Small Blocks! Part 189
Text Ciphers: Let's Mix Small Blocks! Part 190
Text Ciphers: Let's Mix Small Blocks! Part 191
Text Ciphers: Let's Mix Small Blocks! Part 192
Text Ciphers: Let's Mix Small Blocks! Part 193
Text Ciphers: Let's Mix Small Blocks! Part 194
Text Ciphers: Let's Mix Small Blocks! Part 195
Text Ciphers: Let's Mix Small Blocks! Part 196
Text Ciphers: Let's Mix Small Blocks! Part 197
Text Ciphers: Let's Mix Small Blocks! Part 198
Text Ciphers: Let's Mix Small Blocks! Part 199
Text Ciphers: Let's Mix Small Blocks! Part 200
Text Ciphers: Let's Mix Small Blocks! Part 201
Text Ciphers: Let's Mix Small Blocks! Part 202
Text Ciphers: Let's Mix Small Blocks! Part 203
Text Ciphers: Let's Mix Small Blocks! Part 204
Text Ciphers: Let's Mix Small Blocks! Part 205
Text Ciphers: Let's Mix Small Blocks! Part 206
Text Ciphers: Let's Mix Small Blocks! Part 207
Text Ciphers: Let's Mix Small Blocks! Part 208
Text Ciphers: Let's Mix Small Blocks! Part 209
Text Ciphers: Let's Mix Small Blocks! Part 210
Text Ciphers: Let's Mix Small Blocks! Part 211
Text Ciphers: Let's Mix Small Blocks! Part 212
Text Ciphers: Let's Mix Small Blocks! Part 213
Text Ciphers: Let's Mix Small Blocks! Part 214
Text Ciphers: Let's Mix Small Blocks! Part 215
Text Ciphers: Let's Mix Small Blocks! Part 216
Text Ciphers: Let's Mix Small Blocks! Part 217
Text Ciphers: Let's Mix Small Blocks! Part 218
Text Ciphers: Let's Mix Small Blocks! Part 219
Text Ciphers: Let's Mix Small Blocks! Part 220
Text Ciphers: Let's Mix Small Blocks! Part 221
Text Ciphers: Let's Mix Small Blocks! Part 222
Text Ciphers: Let's Mix Small Blocks! Part 223
Text Ciphers: Let's Mix Small Blocks! Part 224
Text Ciphers: Let's Mix Small Blocks! Part 225
Text Ciphers: Let's Mix Small Blocks! Part 226
Text Ciphers: Let's Mix Small Blocks! Part 227
Text Ciphers: Let's Mix Small Blocks! Part 228
Text Ciphers: Let's Mix Small Blocks! Part 229
Text Ciphers: Let's Mix Small Blocks! Part 230
Text Ciphers: Let's Mix Small Blocks! Part 231
Text Ciphers: Let's Mix Small Blocks! Part 232
Text Ciphers: Let's Mix Small Blocks! Part 233
Text Ciphers: Let's Mix Small Blocks! Part 234
Text Ciphers: Let's Mix Small Blocks! Part 235
Text Ciphers: Let's Mix Small Blocks! Part 236
Text Ciphers: Let's Mix Small Blocks! Part 237
Text Ciphers: Let's Mix Small Blocks! Part 238
Text Ciphers: Let's Mix Small Blocks! Part 239
Text Ciphers: Let's Mix Small Blocks! Part 240
Text Ciphers: Let's Mix Small Blocks! Part 241
Text Ciphers: Let's Mix Small Blocks! Part 242
Text Ciphers: Let's Mix Small Blocks! Part 243
Text Ciphers: Let's Mix Small Blocks! Part 244
Text Ciphers: Let's Mix Small Blocks! Part 245
Text Ciphers: Let's Mix Small Blocks! Part 246
Text Ciphers: Let's Mix Small Blocks! Part 247
Text Ciphers: Let's Mix Small Blocks! Part 248
Text Ciphers: Let's Mix Small Blocks! Part 249
Text Ciphers: Let's Mix Small Blocks! Part 250
Text Ciphers: Let's Mix Small Blocks! Part 251
Text Ciphers: Let's Mix Small Blocks! Part 252
Text Ciphers: Let's Mix Small Blocks! Part 253
Text Ciphers: Let's Mix Small Blocks! Part 254
Text Ciphers: Let's Mix Small Blocks! Part 255
Text Ciphers: Let's Mix Small Blocks! Part 256

Terry Ritter

Some block ciphers handle data blocks of dramatically variable size. These are not stream ciphers, but are block ciphers with variable block sizes. Such ciphers may offer different avalanche and avalanche behavior for different block sizes. This paper reviews the literature on avalanche in large block ciphers, which suggests the possibility that mixing could be more effective for such ciphers than for ciphers with small block sizes. This paper also reviews the literature on measuring avalanche, and presents a new method for measuring avalanche in large block ciphers. This method is then used to measure avalanche in Dynamic Substitution Combiners (combiners using Dynamic Substitution) and in Dynamic Substitution Ciphers (ciphers using Dynamic Substitution for mixing). Some of these ciphers show very good avalanche behavior, which suggests that Dynamic Substitution may be a very effective mixing method for large block ciphers.

INTRODUCTION

The simple block cipher is a keyed invertible substitution table of sufficient size. Unfortunately, with 128-bit blocks, there would be 2^128 entries in the table, and this is not practical for any purpose. So modern block ciphers use a "function" to combine a relatively small number of simple operations into a "round" which is then repeated to provide the effectiveness of the substitution technology. In particular, the block cipher is a keyed invertible function of its input.

A more fundamental approach might be "what one would repeat" a large block cipher. The desired answer is that there can be two: one for small blocks and one for large blocks:

1. Large block ciphers can be constructed almost of necessity from "strong" or each-plaintext-element-to-each-ciphertext-element mixing. Such mixing is needed in small block ciphers, but it is not clear that such mixing is needed in large block ciphers. Such mixing in large block ciphers may be a function of the component blocks, or it may be a function of the entire block, or it may be a function of some additional parameter.
2. A large block has room for information other than data. Such information could be used to provide a keyed or data-dependent mixing operation. Such a mixing function could be a simple invertible table, or it could be a function of the data. We can provide ultra-high speed mixing. The same table also can provide a high-speed nonlinear substitution, which may be useful in other ways. Such a table could also be used to provide a keyed or data-dependent substitution.

Note that a cipher which requires a block size of dimensionally "large" size may be a cipher which is not practical for any purpose, but it may be an "impractical" design which is a simple generalized primitive which can be used to study the effects of large block sizes. If such a design is available, and where large blocks are used, we can always "slow down" the cipher by using a "round" function which is a small-block cipher. This is a form of legacy cipher. This means that there is little disadvantage to a large block cipher, and there may be some advantage in being able to do so.

Here we seek to investigate the quality of the mixing in some large block ciphers proposed by Ralph Merkle.

AVALANCHE

The term "avalanche" comes from Feistel [1], and refers to the "cascading" effect of a small change in the input which causes a change in a strong manner in the output. The change of a single bit in the input should cause a change in half the output bits. It is very more but changes after another round, and so on, until about half the output bits have changed. This is a form of "mixing" which is used in a block cipher which uses small blocks and a "round" function.

The term "avalanche" has been strongly associated with the quality of mixing in block ciphers, and it is a very good term for that purpose. It is not the term "randomness" which Knuth uses in a slightly extended sense, or "entropy" which Shannon uses in a very well-defined sense. Avalanche is a very specialized property, and it is not always available, and when large blocks are used, we can always "slow down" the cipher by using a "round" function which is a small-block cipher. This is a form of legacy cipher. This means that there is little disadvantage to a large block cipher, and there may be some advantage in being able to do so.

Here we seek to investigate the quality of the mixing in some large block ciphers proposed by Ralph Merkle.

AVALANCHE

The term "avalanche" comes from Feistel [1], and refers to the "cascading" effect of a small change in the input which causes a change in a strong manner in the output. The change of a single bit in the input should cause a change in half the output bits. It is very more but changes after another round, and so on, until about half the output bits have changed. This is a form of "mixing" which is used in a block cipher which uses small blocks and a "round" function.

> The term avalanche comes from Feistel [1], and refers to the "cascading" effect of a small change in the input which causes a change in a strong manner in the output. The change of a single bit in the input should cause a change in half the output bits. It is very more but changes after another round, and so on, until about half the output bits have changed. This is a form of "mixing" which is used in a block cipher which uses small blocks and a "round" function.

The term avalanche

Measured Boolean Function Nonlinearity in Feistel Cipher Constructions

A *Ciphers By Ritter* Page

Terry Ritter

Nonlinearity is the number of bits which must change in the truth table of a Boolean function to reach the closest affine function, and thus is a measure of one kind of cipher strength. By measuring the nonlinearity of Feistel constructions with various numbers of rounds and tables, we hope to gain insight into how Feistel mixing compares to other alternatives.

From: ritter@io.com (Terry Ritter)
Newsgroups: sci.crypt
Subject: Measured Nonlinearity in Feistel Constructions (LONG!)
Date: Thu, 26 Feb 1998 06:39:28 GMT
Lines: 846
Message-ID: <34f50e19.599891@news.io.com>

MEASURED NONLINEARITY IN FEISTEL CONSTRUCTIONS

Terry Ritter
Ritter Software Engineering
http://www.io.com/~ritter/

1998-02-25

Abstract

Nonlinearity is the number of bits which must change in the truth table of a Boolean function to reach the closest affine function, and thus is a measure of one kind of cipher strength. By measuring the nonlinearity of Feistel constructions with various numbers of rounds and tables, we hope to gain insight into how Feistel mixing compares to other alternatives.

Introduction

The ideal block cipher is a keyed simple substitution table of sufficient size. Unfortunately, with 128-bit blocks, there would be 2**128 entries in that table, which is completely out of the question. So the modern block cipher is a *construction* intended to *simulate* a keyed substitution of the desired size. At issue is the effectiveness of the construction technology. One way to investigate this is by using Boolean function theory, since a substitution table -- or cipher -- can be considered a set of independent Boolean functions, one for each output bit.

A Boolean function produces a single-bit result for each possible combination of values from perhaps many Boolean variables. The *nonlinearity* of a Boolean function is the Hamming distance to the closest affine function [e.g., PIE88, PIE89, PIE89B]. (Also see:

http://www.io.com/~ritter/ARTS/MEASNONL.HTM

). That is, nonlinearity is the number of bits which must change in the truth table of a Boolean function to reach the closest affine function. If "linearity" is considered a significant cryptographic weakness, nonlinearity is an explicit measure of the *lack* of that weakness. So nonlinearity measures one form of cipher "strength."

For cryptographic purposes, it is desired to take the nonlinearity of a substitution table to be the *minimum* of the nonlinearity values for each output bit in that table. Nonlinearity is measured by forming the (one-bit-wide) truth table for a particular output bit, then performing a Fast Walsh-Hadamard Transform (FWT) on that array. Each result value is essentially a correlation count to a particular affine function, and the minimum distance (the maximum correlation) is found by scanning the transform results.

In measuring nonlinearity, it is generally necessary to record the function result for each possible combination of input variables. If we have an "8-bit" table, we must record and then transform 256 elements, and if we have a "16-bit" table, we must record and transform 64K elements. Thus, measuring large functions rapidly becomes impossible. So, although we cannot hope to measure the nonlinearity of a real 64-bit or 128-bit block cipher, we *can* measure nonlinearity in substitution tables and small block constructions. Here we deal with 5-bit tables and the resulting 10-bit blocks.

Ideal Nonlinearity Distributions

By properly shuffling tables using a large-state cryptographic RNG (in particular:

http://www.io.com/~ritter/KEYSHUF.HTM

) we can sample or construct a presumably uniform distribution of different table arrangements. We can measure the nonlinearity of each table, and accumulate the results. For 5-bit tables, we get a nonlinearity distribution something like that shown in Figure 1.

```
  Nonlinearity Distribution in 5-Bit Tables   Fig. 1

   0.7 |              *
   0.6 |              *
   0.5 |              *
   0.4 |              *
   0.3 |              *
   0.2 |           *  *
   0.1 |           *  *  *
   0.0 |           *  *  *
  Prob +--+--+--+--+--+--+--
          0  2  4  6  8 10   Nonlinearity
```

Our block cipher construction problem essentially involves using small random tables (here 5-bit tables with 32 entries and a mean nonlinearity of 7.8) to somehow emulate a table which is twice as wide (here a 10-bit table with 1024 entries and a mean nonlinearity of 447.7).

For 10-bit tables, we get a nonlinearity distribution something like that in Figure 2.

```
  Nonlinearity Distribution in 10-Bit Tables   Fig. 2

  0.2   |
  0.175 |                     *  *
  0.15  |                     *  *  *
  0.125 |                  *  *  *  *
  0.1   |               *  *  *  *  *
  0.075 |               *  *  *  *  *  *
  0.05  |         *  *  *  *  *  *  *  *
  0.025 |      *  *  *  *  *  *  *  *  *  *
  0.00  |   *  *  *  *  *  *  *  *  *  *  *  *
  Prob  +--+--+--+--+--+--+--+--+--+--+--+--+--
          436   440   444   448   452   456   Nonlinearity
```

Feistel Mixing

The classic Feistel construction of Figure 3 is a way of mixing two small blocks into a large block. Here, + is the exclusive-OR function. As used in the U.S. Data Encryption Standard (DES), L and R are each 32 bits; here we have only 5 bits in each.

```
   Feistel Construction   Fig. 3

       L          R
       |          |
       |--> F --> +    round 1
       |          |
       + <-- F <--|    round 2
       |          |
       v          v
       L'         R'
```

In DES, ideal properties for F are much-discussed topics. The Feistel construction itself supports the ability to encipher and decipher using a random function F, not necessarily a permutation. But the 8 different "S-boxes" used in DES *are* permutations, if we see each as a set of 4 permutations per box, so using random permutations in Feistel mixing is not at all unreasonable. And by using permutation tables as components we have a single compatible measure both for the component tables and the simulated larger table, and thus can draw conclusions about the construction. Invertible tables also support a direct comparison to earlier work on nonlinearity with respect to Mixing and VSBC constructions.

Unlike the earlier work, the decision to use random invertible substitutions for function F still leaves us with a couple of remaining parameters: The number of rounds, and the number of tables. Here we investigate the mixing obtained from the Feistel construction with two variations: The use of a single table for various numbers of rounds, and the use of a new random table on every other round, for various numbers of rounds.

Feistel Mixing and DES

Unfortunately, the Feistel construction is *not* the only form of mixing found in DES. Indeed, the practicality of the Feistel construction depends upon having (that is, *constructing*) a wide function F. And while F *might* be realized as a recursive sequence of smaller Feistel constructions, any such structure would be impractically slow. So the very existence of practical Feistel ciphers depends upon the construction of a random function F which itself mixes and diffuses information across its width.

In each DES "S-box," the outside input bits (b0 and b5) -- which are set by expansion "E" -- just select which of the 4 boxes to use. This is a data-dependent dynamic table selection which is, in itself, a form of mixing. Of course, the simple combination of expansion "E" and the array of "S-boxes" by themselves probably would not produce good mixing: A single bit-change at one end of the input block could propagate no faster than one table per round, *if* all data were kept aligned. But this is not the case, since "Permutation P" transposes the data bits. So, other than the dynamic table selection (the two extra inputs for each S-box), function F is itself a classic substitution-permutation design.

This means that the overall mixing in DES is a complex combination of the Feistel construction *plus* the expansion, table-selection, table lookup and permutation in each round. This complexity makes it difficult to independently extract the importance and consequences of each feature, as we surely must if we are to deeply understand the design. This confluence of multiple features also makes it impossible to scale DES to a tiny testable size. Thus, we seem to be limited to testing a tiny Feistel construction -- which *is* scalable -- instead of a tiny DES.

Feistel Mixing with Random Tables

A common assumption about Feistel mixing is that some fixed small set of optimized tables will be used repeatedly. This concept presumably comes from DES. But in the special context of DES:

1. DES S-boxes each contain 4 separate *permutations*, and are *not* random constructions; and

2. The DES permutations are only 4 bits wide, and 4-bit-wide permutations are just about the weakest possible tables that have any nonlinear strength at all.

Accordingly, it should come as no surprise that replacing DES S-boxes with random tables is not a good idea. And random 4-bit permutations are often weak, so this is *also* not a good idea.

In marked contrast, random 8-permutations are almost *never* that weak (the chance of finding even one linear output is 1 in 2**239). It thus appears that the assumption that random tables are necessarily bad is based on the special context of DES, and is not valid in general.

Comparisons to Mixing and VSBC Constructions

Earlier nonlinearity measurement experiments on tiny Mixing (

http://www.io.com/~ritter/ARTS/MIXNONLI.HTM

) and VSBC (

http://www.io.com/~ritter/ARTS/VSBCNONL.HTM

) constructions also used random tables, thus potentially making it possible to see how Feistel mixing compares.

Unfortunately, the comparison is not exact, because both the Mixing and VSBC constructions do in some sense mix multiple tables in a single mixing level. With Feistel mixing, we would normally think of multiple "rounds" of mixing, rather than having several tables in each round. DES does of course manage this, but does not give us a scalable rule which we could use to build a tiny version with multiple tables. And even in the best possible situation we would be looking at the experimental measurement of a 16-bit simulated table, for which it would be very difficult to collect an accurate ideal distribution.

So, given the limitations of reality, the first thing we might want to look at is the nonlinearity measure from Feistel mixing with a single table, under a varying the number of rounds. Then we might look at using multiple tables, again for a varying number of rounds.

The Experiments

We construct 10,000 random 5-bit tables, which are then used in 10-bit Feistel constructions. Each of these constructions is measured for nonlinearity, and the result accumulated.

While Figure 2 gives a general idea of the desired distribution, we eventually want to use a chi-square comparison, for which we need an ideal reference distribution. The ideal model in Figure 4 is thus constructed from three different 15-hour trials of 1,000,000 random 10-bit tables each, with the counts reduced by a factor of 100 and rounded to an integer.

The Ideal 10-Bit Nonlinearity Distribution   Fig. 4

NL	Trial 1	Trial 2	Trial 3	Ideal
462	0	1	0	0
460	55	75	55	1
458	2145	2269	2234	22
456	20537	20938	20882	208
454	76259	76344	75980	763
452	148847	148135	148510	1485
450	188329	187802	187585	1878
448	177775	178208	178616	1782
446	140283	140219	140407	1403
444	97415	97317	97465	974
442	61999	62282	62213	622
440	37829	37685	37382	376
438	21675	21789	21753	218
436	12417	12320	12408	124
434	6826	6818	6912	68
432	3740	3693	3584	37
430	1912	2050	1884	19
428	979	1043	1073	10
426	490	537	526	5
424	249	241	265	2
422	119	133	143	1
420	64	55	69	0
418	26	25	30	0
416	12	14	12	0
414	9	5	6	0
412	3	2	4	0
410	0	0	1	0

In each experimental trial, we first initialize the substitutions from a random key. This produces a particular 10-bit cipher -- a simulated 10-bit simple substitution. We then traverse the complete transformation -- all 1024 possible data values -- and collect all 1024 10-bit ciphertext results. We then traverse all 10 ciphertext bit-columns, one-by-one, and perform a nonlinearity computation on each. The minimum nonlinearity value found across all columns is accumulated into the growing distribution.

We see the results for 2 rounds of Feistel mixing in Figure 5.

Feistel Nonlinearity Distribution
One Table, Two Rounds   Fig. 5

NL	Expect	Found
64	0	2
128	0	78
192	0	1494
256	0	7387
320	0	1039

Clearly, 2 Feistel mixing rounds do not approximate the nonlinearity distribution from random substitutions, so we move on to 4 rounds as in Figure 6.

Feistel Nonlinearity Distribution
One Table, Four Rounds   Fig. 6

NL	Expect	Found
224	0	1
312	0	56
320	0	1
344	0	1
352	0	35
364	0	1
368	0	8
384	0	696
388	0	3
392	0	187
396	0	2
400	0	11
404	0	15
406	0	3
408	0	13
410	0	2
412	0	38
414	0	2
416	0	1709
418	0	4
420	0	66
422	1	11
424	2	126
426	5	13
428	10	256
430	19	20
432	37	780
434	68	51
436	124	660
438	218	109
440	376	3566
442	622	70
444	974	589
446	1403	91
448	1782	639
450	1878	43
452	1485	107
454	763	12
456	208	2
458	22	1
460	1	0
462	0	0

Again, the is pretty brutal: So even four Feistel mixing rounds do not approximate random substitutions. And neither do six (Figure 7), eight (Figure 8), ten (Figure 9) or twelve (Figure 10):

Feistel Nonlinearity Distribution
One Table, Six Rounds   Fig. 7

NL	Expect	Found
296	0	1
316	0	1
324	0	1
332	0	1
336	0	2
352	0	1
372	0	1
384	0	2
388	0	3
392	0	1
394	0	1
396	0	2
398	0	1
400	0	3
402	0	1
404	0	10
406	0	0
408	0	23
410	0	2
412	0	22
414	0	6
416	0	22
418	0	14
420	0	46
422	1	25
424	2	85
426	5	21
428	10	114
430	19	70
432	37	192
434	68	138
436	124	386
438	218	269
440	376	641
442	622	662
444	974	1197
446	1403	1238
448	1782	1612
450	1878	1394
452	1485	1151
454	763	492
456	208	133
458	22	13
460	1	0
462	0	0

Feistel Nonlinearity Distribution
One Table, Eight Rounds   Fig. 8

NL	Expect	Found
346	0	1
370	0	1
372	0	1
402	0	2
404	0	3
406	0	3
408	0	6
410	0	1
412	0	6
414	0	10
416	0	17
418	0	18
420	0	30
422	1	36
424	2	38
426	5	53
428	10	74
430	19	95
432	37	137
434	68	194
436	124	261
438	218	406
440	376	519
442	622	808
444	974	1080
446	1403	1365
448	1782	1569
450	1878	1527
452	1485	1063
454	763	519
456	208	136
458	22	21
460	1	0
462	0	0

Feistel Nonlinearity Distribution
One Table, Ten Rounds   Fig. 9

NL	Expect	Found
362	0	1
396	0	1
398	0	2
400	0	1
402	0	1
404	0	0
406	0	4
408	0	3
410	0	10
412	0	10
414	0	11
416	0	28
418	0	16
420	0	25
422	1	39
424	2	63
426	5	63
428	10	78
430	19	116
432	37	155
434	68	247
436	124	314
438	218	380
440	376	584
442	622	811
444	974	1034
446	1403	1353
448	1782	1556
450	1878	1428
452	1485	1046
454	763	480
456	208	126
458	22	14
460	1	0
462	0	0

Feistel Nonlinearity Distribution
One Table, Twelve Rounds   Fig. 10

NL	Expect	Found
398	0	1
400	0	3
402	0	2
404	0	2
406	0	6
408	0	4
410	0	8
412	0	12
414	0	13
416	0	12
418	0	30
420	0	23
422	1	34
424	2	64
426	5	62
428	10	96
430	19	113
432	37	179
434	68	234
436	124	327
438	218	462
440	376	597
442	622	797
444	974	1054
446	1403	1347
448	1782	1467
450	1878	1450
452	1485	1011
454	763	450
456	208	128
458	22	12
460	1	0
462	0	0

The twelve rounds with a fixed table of Figure 10 is about as good as it gets. The distribution from a single table never gets close enough to the ideal distribution to make a chi-square computation reasonable.

Feistel Mixing with Random Multiple Tables

On the other hand, things improve if we have a new table every couple of rounds (Figure 11). (There would seem to be little advantage in having a unique table for each round, since only half of the block would be affected by that table.)

Feistel Nonlinearity Distribution
Two Tables, Four Rounds   Fig. 11

NL	Expect	Found
320	0	3
352	0	39
360	0	2
364	0	1
368	0	6
382	0	1
384	0	156
386	0	1
388	0	3
390	0	0
392	0	225
394	0	1
396	0	4
398	0	1
400	0	6
402	0	1
404	0	13
406	0	3
408	0	22
410	0	4
412	0	42
414	0	2
416	0	1951
418	0	3
420	0	81
422	1	5
424	2	143
426	5	13
428	10	276
430	19	29
432	37	853
434	68	47
436	124	788
438	218	90
440	376	3175
442	622	98
444	974	805
446	1403	130
448	1782	833
450	1878	33
452	1485	95
454	763	10
456	208	5
458	22	1
460	1	0
462	0	0

While hardly ideal, we can see that the results in Figure 11 are somewhat better than the same number of rounds with a single table (Figure 6). But things perk up rather nicely with three tables and six rounds as shown in Figure 12.

Feistel Nonlinearity Distribution
Three Tables, Six Rounds   Fig. 12

NL	Expect	Found		Chi-Sq	DF
420	0	0	}		
422	1	3	}		
424	2	3	}		
426	5	3	}		
428	10	10	}	0.056	0
430	19	13		1.950	1
432	37	40		2.194	2
434	68	59		3.385	3
436	124	123		3.393	4
438	218	200		4.879	5
440	376	381		4.946	6
442	622	633		5.140	7
444	974	990		5.403	8
446	1403	1390		5.523	9
448	1782	1780		5.526	10
450	1878	1952		8.441	11
452	1485	1441		9.745	12
454	763	754		9.851	13
456	208	208		9.851	14
458	22	17	}		
460	1	0	}		
462	0	0	}	11.417	15

With 15 "degrees of freedom," the chi-square value of 11.417 in Figure 11 is very believable. The critical chi-square values for DF = 15 are given in Figure 13.

Chi-Square Critical Values, for DF = 15   Fig. 13

1%	5%	25%	50%	75%	95%	99%
5.2293	7.2609	11.0365	14.3389	18.2451	24.9958	30.5779

And another couple of rounds with yet another table continues the success story in Figure 14.

Feistel Nonlinearity Distribution
Four Tables, Eight Rounds   Fig. 14

NL	Expect	Found		Chi-Sq	DF
410	0	1	}		
412	0	0	}		
414	0	0	}		
416	0	0	}		
418	0	1	}		
420	0	2	}		
422	1	2	}		
424	2	1	}		
426	5	4	}		
428	10	10	}	0.500	0
430	19	19		0.500	1
432	37	30		1.824	2
434	68	65		1.957	3
436	124	119		2.158	4
438	218	222		2.232	5
440	376	375		2.234	6
442	622	634		2.466	7
444	974	982		2.532	8
446	1403	1365		3.561	9
448	1782	1750		4.135	10
450	1878	1928		5.467	11
452	1485	1504		5.710	12
454	763	750		5.931	13
456	208	212		6.008	14
458	22	23	}		
460	1	1	}		
462	0	0	}	6.052	15

While the chi-square value of 6.052 may seem small, this means the distribution is close to the ideal. When comparing distributions, we are rarely worried about a chi-square value that seems "too good," for the alternative is far *less* believable: Either we have had the good luck of sampling a close distribution so it looks better than it should, or we have had the absolutely unbelievable luck of sampling a fundamentally different distribution so it looks not just good, but actually *too* good. Thus, these tests are basically "one-tail" statistical tests.

Conclusions

Feistel mixing seeks to produce a block cipher which is double the size of a smaller "block cipher" component; in this respect, it is similar to the Mixing constructions measured earlier. When using a single fixed table, it appears that Feistel mixing is never good enough, no matter how many rounds are applied. But even 6 rounds of Feistel mixing with 3 *different* tables apparently does approximate the nonlinearity of a larger substitution table.

These experiments show an ultimately successful result using random invertible tables. Thus, the experiments give no indication that Feistel mixing requires the ideal tables that so much of the literature seems directed at producing. In contrast to the unending search for the qualities of an ideal table, it appears that we can produce an effective Feistel cipher by choosing tables "at random" (that is, we can construct tables based on the cipher key). Note that Differential and Linear Cryptanalysis style attacks generally depend upon knowledge of the contents of the tables, knowledge which is unavailable when keyed tables are used.

When compared to the Mixing and VSBC constructions investigated previously, the problem with Feistel mixing is that, by itself, it only doubles the size of the nonlinear component, and we assume that repeated doubling requires exponential effort: That is, if 6 Feistel rounds are required to take random 8-bit tables to a 16-bit emulated table, 6 rounds of the emulation (36 total rounds) should be required to get a 32-bit table, and 216 total rounds should be required for a 64-bit cipher. (It would be interesting to check these assumptions on modern fast equipment.)

Now, we know that DES (an emulated 64-bit table) does *not* require exponential effort. But Feistel mixing is *not* the only form of mixing in DES. We avoid addressing the other structures, because they do not scale well, and because we wish to examine each contribution as independently as possible.

Here we have measured the Feistel mixing alone, which we assume requires effort exponential with the block size. In contrast, Mixing constructions require only effort proportional to n log n, and VSBC constructions require only effort linear with block size. Thus, these alternative constructions offer the possibility of far more efficient cryptographic mixing.

References and Bibliography

[AY82] Ayoub, F. 1982. Probabilistic completeness of substitution-permutation encryption networks. IEE Proceedings, Part E. 129(5): 195-199.

[DAE94] Daemen, J., R. Govaerts and J. Vandewalle. 1994. Correlation Matrices. Fast Software Encryption. 275-285.

[FOR88] Forre, R. 1988. The Strict Avalanche Criterion: Spectral Properties of Boolean Functions and an Extended Definition. Advances in Cryptology -- CRYPTO '88. 450-468.

[GOR82] Gordon, J. and H. Retkin. 1982. Are Big S-Boxes Best? Cryptography. Proceedings of the Workshop on Cryptography, Burg Feuerstein, Germany, March 29-April 2, 1982. 257-262.

[HEY94] Heys, H. and S. Tavares. 1994. On the security of the CAST encryption algorithm. Canadian Conference on Electrical and Computer Engineering. Halifax, Nova Scotia, Canada, Sept. 1994. 332-335.

[HEY95] Heys, H. and S. Tavares. 1995. Known plaintext cryptanalysis of tree-structured block ciphers. Electronics Letters. 31(10): 784-785.

[MEI89] Meier, W. and O. Staffelbach. 1989. Nonlinearity Criteria for Cryptographic Functions. Advances in Cryptology -- Eurocrypt '89. 549-562.

[MIR97] Mirza, F. 1997. Linear and S-Box Pairs Cryptanalysis of the Data Encryption Standard.

[OC91] O'Connor, L. 1991. Enumerating nondegenerate permutations. Advances in Cryptology -- Eurocrypt '91. 368-377.

[OC93] O'Connor, L. 1993. On the Distribution Characteristics in Bijective Mappings. Advances in Cryptology -- EUROCRYPT '93. 360-370.

[PIE88] Pieprzyk, J. and G. Finkelstein. 1988. Towards effective nonlinear cryptosystem design. IEE Proceedings, Part E. 135(6): 325-335.

[PIE89] Pieprzyk, J. and G. Finkelstein. 1989. Permutations that Maximize Non-Linearity and Their Cryptographic Significance. Computer Security in the Age of Information. 63-74.

[PIE89B] Pieprzyk, J. 1989. Non-linearity of Exponent Permutations. Advances in Cryptology -- EUROCRYPT '89. 80-92.

[PIE93] Pieprzyk, J., C. Charnes and J. Seberry. 1993. Linear Approximation Versus Nonlinearity. Proceedings of the Workshop on Selected Areas in Cryptography (SAC '94). 82-89.

[PRE90] Preneel, B., W. Van Leekwijck, L. Van Linden, R. Govaerts and J. Vandewalle. 1990. Propagation Characteristics of Boolean Functions. Advances in Cryptology -- Eurocrypt '90. 161-173.

[RUE86] Rueppel, R. 1986. Analysis and Design of Stream Ciphers. Springer-Verlag.

[XIO88] Xiao, G-Z. and J. Massey. 1988. A Spectral Characterization of Correlation-Immune Combining Functions. IEEE Transactions on Information Theory. 34(3): 569-571.

[YOU95] Youssef, A. and S. Tavares. 1995. Resistance of Balanced S-boxes to Linear and Differential Cryptanalysis. Information Processing Letters. 56: 249-252.

[YOU95B] Youssef, A. and S. Tavares. 1995. Number of Nonlinear Regular S-boxes. Electronics Letters. 31(19): 1643-1644.

[ZHA95] Zhang, X. and Y. Zheng. 1995. GAC -- the Criterion for Global Avalanche Characteristics of Cryptographic Functions. Journal for Universal Computer Science. 1(5): 316-333.

Terry Ritter   ritter@io.com   http://www.io.com/~ritter/
The Crypto Glossary: http://www.io.com/~ritter/GLOSSARY.HTM

Also see: **Measured Nonlinearity in Variable Size Block Ciphers (1998)** (29K), **Measuring Nonlinearity by Walsh Transform (1998)** (20K), *and* **Measured Nonlinearity in Mixing Constructions (1997)** (25K).

Terry Ritter, his current address, and his top page.

Last updated: 1998-02-26

Measuring Boolean Function Nonlinearity by Walsh Transform

A *Ciphers By Ritter* Page

Terry Ritter

Nonlinearity is the number of bits which must change in the truth table of a Boolean function to reach the closest affine function. A basic background in affine Boolean functions and their relationship to Walsh-Hadamard functions is presented. It is shown how distances to linear functions can be computed by hand, and that these values are exactly the same as those produced by a Fast Walsh Transform (FWT), which also can be computed by hand. A Pascal routine for the FWT is presented as a basis for practical nonlinearity measurement.

From: ritter@io.com (Terry Ritter)
Newsgroups: sci.crypt
Subject: Measuring Nonlinearity By Walsh Transform (Long!)
Date: Tue, 13 Jan 1998 05:10:08 GMT
Lines: 522
Message-ID: <34baf5ee.192497@news.io.com>

MEASURING NONLINEARITY BY WALSH TRANSFORM

Terry Ritter   ritter@io.com
Ritter Software Engineering
http://www.io.com/~ritter/

1998-01-12

Abstract

Nonlinearity is the number of bits which must change in the truth table of a Boolean function to reach the closest affine function. A basic background in affine Boolean functions and their relationship to Walsh-Hadamard functions is presented. It is shown how distances to linear functions can be computed by hand, and that these values are exactly the same as those produced by a Fast Walsh Transform (FWT), which also can be computed by hand. A Pascal routine for the FWT is presented as a basis for practical nonlinearity measurement.

Affine Boolean Functions

A Boolean function produces a single-bit result for each possible combination of values from perhaps many Boolean variables. The Boolean field consists of the values {0,1}, with XOR as "addition" and AND as "multiplication."

We call a function "constant" when it has only one fixed value, which we often denote "k":

(1) f = k

I call a function "linear" when it fits the plane geometry equation of a line (here the constant is "b"):

(2) y = mx + b

Note that in the Boolean field, a constant value of '1' acts to reverse the sense (invert) the output value, while a constant of '0' does not affect the result at all.

I call a function "affine" when it fits the form:

(3) $f = a_{n-1} * x_{n-1} + a_{n-2} * x_{n-2} + ... + a_1 * x_1 + a_0$

(Others reserve "affine" for Equation 3 *without* a constant, and use the term "linear" for Equation 3 *with* a constant.)

In Boolean functions, each of the coefficients a[i] acts only to *select* each different variable x[i], and if we consider all a[i] a number, we have a unique ordering for affine Boolean functions as shown in Figure 1.

```
   Affine Functions   Fig. 1

      f0 = 0
      f1 = 1
      f2 = x[1]
      f3 = x[1] + 1
      f4 = x[2]
      f5 = x[2] + 1
      f6 = x[2] + x[1]
      f7 = x[2] + x[1] + 1
       . . .
```

Thus, we can write 16 different forms for 3 variables. But it is convenient to pair the functions which are the same except for the value of the constant, and then we have exactly 8 affine Boolean functions of 3 variables (with and without constant). Of course, any possible Boolean function of 3 variables *also* can be expressed in 8 terms, one for every possible combination of variable value. This is the "truth table" for a function, or the function expressed as the sequence of its output values.

We can now express every possible 3-variable affine Boolean function, in order, as shown in Figure 2.

```
   The 3-Variable Affine Boolean Functions   Fig. 2

     affine    truth table

          1    1  1  1  1  1  1  1  1
         x0    0  1  0  1  0  1  0  1
      x1       0  0  1  1  0  0  1  1
      x1+x0    0  1  1  0  0  1  1  0
   x2          0  0  0  0  1  1  1  1
   x2+   x0    0  1  0  1  1  0  1  0
   x2+x1       0  0  1  1  1  1  0  0
   x2+x1+x0    0  1  1  0  1  0  0  1
```

Correlation

One way to measure a sort of "correlation" between two Boolean functions is to compare their truth tables and count the number of bits which differ; this is their Hamming distance. Then, since we expect about half the bit positions to differ (on average), we can subtract the expected difference: this gives what I am calling -- for lack of a better term -- "unexpected distance" (UD). The magnitude of the UD relates to how unexpected the distance is, while the sign indicates the direction. Consider two functions and their difference as shown in Figure 3:

```
   Distance to an Affine Function   Fig. 3

           f    1  0  0  1  1  1  0  0
   x2+x1+x0    0  1  1  0  1  0  0  1
       diff    1  1  1  1  0  1  0  1
```

We have a Hamming distance of 6 between these two functions. This is an unexpected distance of 6 - 4 = +2, which means that 2 more bits differ than we would expect.

With some work, we can now compare a Boolean function to each possible affine Boolean function, and develop a distance to each. The less distance to an affine function, the more "linear" the measured function must be (see Figure 4).

```
   Distance to Each 3-Variable Affine Boolean Function   Fig. 4

     affine    truth table                 distance

          1    1  1  1  1  1  1  1  1      4
         x0    0  1  0  1  0  1  0  1      4
      x1       0  0  1  1  0  0  1  1      6
      x1+x0    0  1  1  0  0  1  1  0      6
   x2          0  0  0  0  1  1  1  1      4
   x2+   x0    0  1  0  1  1  0  1  0      4
   x2+x1       0  0  1  1  1  1  0  0      2
   x2+x1+x0    0  1  1  0  1  0  0  1      6

          f    1  0  0  1  1  1  0  0
```

So the minimum distance from f to an affine function is 2, making the closest direct function x2+x1. *But*, if we complement those functions which have a distance of 6 (that is, add a constant '1' term in the affine equation) we get the values shown in Figure 5.

```
   Distance to Complement Affine Functions   Fig. 5

               complement table            distance

      x1       1  1  0  0  1  1  0  0      2
      x1+x0    1  0  0  1  1  0  0  1      2
   x2+x1+x0    1  0  0  1  0  1  1  0      2

          f    1  0  0  1  1  1  0  0
```

This *could* mean that we need to consider the distance from f to the closest affine *complement* as well as the non-complement. But this is unnecessary if we use the absolute value of the unexpected distance, as shown in Figure 6.

```
   Unexpected Distance to Affine Boolean Function   Fig. 6

     affine    truth table                 distance    ud

          1    1  1  1  1  1  1  1  1      4           0
         x0    0  1  0  1  0  1  0  1      4           0
      x1       0  0  1  1  0  0  1  1      6          +2
      x1+x0    0  1  1  0  0  1  1  0      6          +2
   x2          0  0  0  0  1  1  1  1      4           0
   x2+   x0    0  1  0  1  1  0  1  0      4           0
   x2+x1       0  0  1  1  1  1  0  0      2          -2
   x2+x1+x0    0  1  1  0  1  0  0  1      6          +2

          f    1  0  0  1  1  1  0  0
```

When we have a balanced function f, the distances to affine functions are always *even*. This is because the affine functions themselves are balanced, and if we change one bit in a balanced function, we must change at least one other bit to keep it balanced. If the first change increased the distance by 1, the next bit change can either increase it again by 1, or return to the original value. Balanced functions always will be at even distances from each other. This issue will arise again in the nonlinearity of the balanced Boolean functions which occur in invertible substitutions.

The Walsh-Hadamard Transform

A Hadamard matrix H is an n x n matrix with all entries +1 or -1, such that all rows are orthogonal and all columns are orthogonal (see, for example, [HED78]).

The usual development (see, for example [SCH87]) starts with a defined 2 x 2 Hadamard matrix H2 which is ((1,1),(1,-1)). Each step consists of multiplying each element in H2 by the previous matrix, thus negating all elements in the bottom-right entry, as shown in Figure 7.

```
        Hadamard Matrix Development   Fig. 7

   H2 = |  1  1 |    H4 = H2 * H2 = | H2  H2 |
        |  1 -1 |                   | H2 -H2 |

   H4 = | | 1  1 |  | 1  1 | | = | 1  1  1  1 |
        | | 1 -1 |  | 1 -1 | |   | 1 -1  1 -1 |
        |                    |   | 1  1 -1 -1 |
        | | 1  1 |  |-1 -1 | |   | 1 -1 -1  1 |
        | | 1 -1 |  |-1  1 | |

   H8 = | H4  H4 | = | 1  1  1  1  1  1  1  1 |
        | H4 -H4 |   | 1 -1  1 -1  1 -1  1 -1 |
                     | 1  1 -1 -1  1  1 -1 -1 |
                     | 1 -1 -1  1  1 -1 -1  1 |
                     | 1  1  1  1 -1 -1 -1 -1 |
                     | 1 -1  1 -1 -1  1 -1  1 |
                     | 1  1 -1 -1 -1 -1  1  1 |
                     | 1 -1 -1  1 -1  1  1 -1 |
```

Now compare H8 from this strange Hadamard development to the affine functions from Figure 2, shown again as Figure 8:

```
   The 3-Variable Affine Boolean Functions   Fig. 8

          c    0  0  0  0  0  0  0  0
         x0    0  1  0  1  0  1  0  1
      x1       0  0  1  1  0  0  1  1
      x1+x0    0  1  1  0  0  1  1  0
   x2          0  0  0  0  1  1  1  1
   x2+   x0    0  1  0  1  1  0  1  0
   x2+x1       0  0  1  1  1  1  0  0
   x2+x1+x0    0  1  1  0  1  0  0  1
```

So if we map the values in the affine truth table: {0,1} -> {1,-1}, we find *the* *same* *functions* as in the Hadamard development. These are the Walsh functions, and here both developments produce the same order, which is called "natural" or "Hadamard." (There are two other canonical orderings for Walsh functions, which we need not get into here.) Walsh functions have fast transforms which reduce the cost of correlation computations from n*n to n log n, a very substantial reduction.

A Fast Walsh Transform

It is easy to do a Fast Walsh Transform by hand. (Well, I say "easy," then always struggle when I actually do it.) Let's do the FWT of function f: (1 0 0 1 1 1 0 0): First note that f has a binary power length, as required. Next, each pair of elements is modified by an "in-place butterfly"; that is, the values in each pair produce two results which replace the original pair, wherever they were originally located. The left result will be the two values added; the right will be the first less the second. That is,

(4) (a',b') = (a+b, a-b)

So for the values (1,0), we get (1+0, 1-0) which is just (1,1). We start out pairing adjacent elements, then every other element, then every 4th element, and so on until the correct pairing is impossible, as shown in Figure 8.

```
   An 8-Element Fast Walsh Transform (FWT)   Fig. 8

   original      1   0   0   1   1   1   0   0
                 ^---^   ^---^   ^---^   ^---^

      first      1   1   1  -1   2   0   0   0
                 ^-------^       ^-------^
                     ^-------^       ^-------^

     second      2   0   0   2   2   0   2   0
                 ^---------------^
                     ^---------------^
                         ^---------------^
                             ^---------------^

      final      4   0   2   2   0   0  -2   2
```

Now compare these results to the UD values we found in Fig. 4:

```
   Unexpected Distance to the Affine Functions   Fig. 9

     affine     ud

          1     0
         x0     0
      x1       +2
      x1+x0    +2
   x2           0
   x2+   x0     0
   x2+x1       -2
   x2+x1+x0    +2
```

Note that all FWT elements -- after the zeroth -- map the U.D. results *exactly* in both magnitude and sign, *and* in the exact same order. (This ordering means that the binary index of any result is also the recipe for expressing the affine function being compared in that position, as shown in Figure 2.) The zeroth element in the FWT (here 4) is a special case and is the number of 1-bits in the function when we use the real values {0,1} to represent the function. When measuring balanced functions of a particular size, the zeroth entry should be obvious.

Thus, to find the distance from any balanced function to the closest affine function, compute the FWT of the given function expressed as the real values {0,1}, then find the *maximum* absolute value of all unexpected distance elements past the zeroth element.

A Fast Walsh Transform Routine

The Fast Walsh Transform by hand is automated in the Borland Pascal listing of Figure 10.

A Fast Walsh Transform (FWT) in Pascal   Fig. 10

```
   TYPE
      Lwd = LongInt;
      LintArray = ARRAY[ 0..16380 ] of LONGINT;


   PROCEDURE LintHadFmSeqWalsh( VAR DatLintAr; lastel: Lwd );
      { Hadamard Walsh from sequential data, in-place }
      VAR
         Dat: LintArray ABSOLUTE DatLintAr;
         a, b: LONGINT;
         stradwid,  { distance between pair of elements }
         blockstart, block, pair, el1, el2: Lwd;
      BEGIN
      stradwid := 1;
      blockstart := lastel;
         REPEAT
         el1 := 0;
         blockstart := blockstart DIV 2;
         FOR block := blockstart DOWNTO 0 DO
            BEGIN
            el2 := el1 + stradwid;
            FOR pair := 0 TO PRED(stradwid) DO
               BEGIN
               a := Dat[ el1 ];
               b := Dat[ el2 ];
               Dat[ el1 ] := a + b;
               Dat[ el2 ] := a - b;
               Inc( el1 );  Inc( el2 );
               END;
            el1 := el2;
            END;
         stradwid := (stradwid + stradwid) AND lastel;
         UNTIL (stradwid = 0);
      END; {LintHadFmSeqWalsh}
```

LintHadFmSeqWalsh takes an array of 32-bit integers, and changes the array data into the Walsh-Hadamard transform of that data. For nonlinearity measures, the input data are {0,1} or {1,-1}; the results are potentially bipolar in either case. (The "lastel" parameter is the last index in the data array which starts at index 0; it is thus always 2**n - 1 for some n. The ABSOLUTE attribute forces Borland Pascal to treat the parameter as a LongInt array of arbitrary size.)

Nonlinearity

Nonlinearity is the number of bits which must change in the truth table of a Boolean function to reach the closest affine function.

With the FWT we compute the "unexpected distance," or distance away from the expected difference. By finding the largest absolute value, we can find the closest "linear" function. This is, in a sense, a *linearity* measure. But we want the *non* linearity, and this is half the number of bits in the function, less the absolute value of the unexpected distance. So we can scan the FWT results either for the maximum absolute value first, or calculate the nonlinearity for each, and take the minimum. Nonlinearity is always positive, and also even if we have a balanced function.

It is common to consider a Boolean function as consisting of the real values {0,1}, but it is *also* common to use the transformation

(5) $x' = -1^x$

where x is {0,1}. This transforms {0,1} -> {1,-1}, although we can do the exact same thing with:

(6) x' = 1 - 2x

This transformation has some implications: Using real values {1,-1} doubles the magnitude and changes the sign of the FWT results, but can simplify nonlinearity for unbalanced functions, because the zeroth term need not be treated specially. But if the Boolean function is balanced, as it will be in the typical invertible substitution table, the zeroth element need not be used at all, so using real values {1,-1} seems to provide no particular benefit in this application.

Summary

The basic concepts of affine Boolean functions have been presented, and a correspondence to Walsh-Hadamard functions illustrated. A Fast Walsh Transform was introduced and explained. Hopefully this will help those seeking to measure nonlinearity, and so help make this measurement better understood and more widely used.

References and Bibliography

[AY82] Ayoub, F. 1982. Probabilistic completeness of substitution-permutation encryption networks. IEE Proceedings, Part E. 129(5): 195-199.

[DAE94] Daemen, J., R. Govaerts and J. Vandewalle. 1994. Correlation Matrices. Fast Software Encryption. 275-285.

[FOR88] Forre, R. 1988. The Strict Avalanche Criterion: Spectral Properties of Boolean Functions and an Extended Definition. Advances in Cryptology -- CRYPTO '88. 450-468.

[GOR82] Gordon, J. and H. Retkin. 1982. Are Big S-Boxes Best? Cryptography. Proceedings of the Workshop on Cryptography, Burg Feuerstein, Germany, March 29-April 2, 1982. 257-262.

[HED78] Hedayat, A. and W. Wallis. 1978. Hadamard Matrices and their Applications. The Annals of Statistics. 6(6): 1184-1238.

[HEY94] Heys, H. and S. Tavares. 1994. On the security of the CAST encryption algorithm. Canadian Conference on Electrical and Computer Engineering. Halifax, Nova Scotia, Canada, Sept. 1994. 332-335.

[HEY95] Heys, H. and S. Tavares. 1995. Known plaintext cryptanalysis of tree-structured block ciphers. Electronics Letters. 31(10): 784-785.

[MEI89] Meier, W. and O. Staffelbach. 1989. Nonlinearity Criteria for Cryptographic Functions. Advances in Cryptology -- Eurocrypt '89. 549-562.

[MIR97] Mirza, F. 1997. Linear and S-Box Pairs Cryptanalysis of the Data Encryption Standard.

[OC91] O'Connor, L. 1991. Enumerating nondegenerate permutations. Advances in Cryptology -- Eurocrypt '91. 368-377.

[OC93] O'Connor, L. 1993. On the Distribution Characteristics in Bijective Mappings. Advances in Cryptology -- EUROCRYPT '93. 360-370.

[PIE88] Pieprzyk, J. and G. Finkelstein. 1988. Towards effective nonlinear cryptosystem design. IEE Proceedings, Part E. 135(6): 325-335.

[PIE89] Pieprzyk, J. and G. Finkelstein. 1989. Permutations that Maximize Non-Linearity and Their Cryptographic Significance. Computer Security in the Age of Information. 63-74.

[PIE89B] Pieprzyk, J. 1989. Non-linearity of Exponent Permutations. Advances in Cryptology -- EUROCRYPT '89. 80-92.

[PIE93] Pieprzyk, J., C. Charnes and J. Seberry. 1993. Linear Approximation Versus Nonlinearity. Proceedings of the Workshop on Selected Areas in Cryptography (SAC '94). 82-89.

[PRE90] Preneel, B., W. Van Leekwijck, L. Van Linden, R. Govaerts and J. Vandewalle. 1990. Propagation Characteristics of Boolean Functions. Advances in Cryptology -- Eurocrypt '90. 161-173.

[RUE86] Rueppel, R. 1986. Analysis and Design of Stream Ciphers. Springer-Verlag.

[SCH86] Schroeder, M. 1986. Number Theory in Science and Communications. Springer-Verlag.

[SCH87] Schroeder, M. and N. Sloane. 1987. New Permutation Codes Using Hadamard Unscrambling. IEEE Transactions on Information Theory. IT-33(1): 144-145.

[XIO88] Xiao, G-Z. and J. Massey. 1988. A Spectral Characterization of Correlation-Immune Combining Functions. IEEE Transactions on Information Theory. 34(3): 569-571.

[YOU95] Youssef, A. and S. Tavares. 1995. Resistance of Balanced S-boxes to Linear and Differential Cryptanalysis. Information Processing Letters. 56: 249-252.

[YOU95B] Youssef, A. and S. Tavares. 1995. Number of Nonlinear Regular S-boxes. Electronics Letters. 31(19): 1643-1644.

[ZHA95] Zhang, X. and Y. Zheng. 1995. GAC -- the Criterion for Global Avalanche Characteristics of Cryptographic Functions. Journal for Universal Computer Science. 1(5): 316-333.

Many Walsh-Hadamard references are available in my Walsh-Hadamard literature review:

http://www.io.com/~ritter/RES/WALHAD.HTM

Terry Ritter   ritter@io.com   http://www.io.com/~ritter/

The Crypto Glossary: http://www.io.com/~ritter/GLOSSARY.HTM

Terry Ritter, his *current address*, and his *top page*.

Last updated: 1998-01-15

Chi-Square Bias in Runs-Up/Down RNG Tests

A *[Ciphers By Ritter](#)* Page

Terry Ritter

When (pseudo) random number generators (RNGs) are tested under a "runs up" or "runs down" test, the expectation values described in Knuth II may not be appropriate. Those expectation values seem designed for RNG's which produce a huge number of different values, and are not accurate for RNG's which generate only a small population of values. These unreasonable expectations can produce the appearance of a problem, even where no problem exists.

Path: news.io.com!not-for-mail
From: ritter@io.com (Terry Ritter)
Newsgroups: sci.crypt
Subject: Chi-Square Bias in Runs-Up/Down RNG Tests
Date: Mon, 30 Jun 1997 05:04:23 GMT
Organization: Illuminati Online
Lines: 304
Message-ID: <33b73dff.2257810@news.io.com>
NNTP-Posting-Host: dialup-02-014.austin.io.com
X-Newsreader: Forte Free Agent 1.11/32.235
Xref: news.io.com sci.crypt:68609

INTRODUCTION

When (pseudo) random number generators (RNGs) are tested under a "runs up" or "runs down" test, the expectation values described in Knuth II may not be appropriate. Those expectation values seem designed for RNG's which produce a huge number of different values, and are not accurate for RNG's which generate only a small population of values. These unreasonable expectations can produce the appearance of a problem, even where no problem exists.

BACKGROUND

Random number generators (RNG's) are state machines designed to produce values which pass statistical tests of randomness. One of the better such tests, surprisingly, is the "runs up" test, described in Knuth II [1] starting on p. 65. Basically, we take a sample value, whereupon we have a "run" of length 1. Then we take another sample, and if this is larger (smaller) than the previous value, we have extended a run up (down) by one count. Eventually the run "fails" and we increment a count for the previous length. Typically, we use chi-square to compare each accumulated length count to the expected value. But this implies that we *know* the expected value.

Knuth II p. 65 refers us to exercise 14 (p. 74) which is the statistically simpler form where a value is discarded between each run. This in turn refers to the "Answers to Exercises" (p. 536) which gives the probability of each run length as:

```
   1/r! - 1/(r+1)!
```

The first term is the probability of getting the given run length *or more*, and the second is the probability for the next higher run length *or more*. Subtracting the second from the first gives us the probability of getting a run of *exactly* length r.

After examining figure (9) on p. 65 it is easy to imagine that runs up (down) testing -- and the above formula -- apply directly to small populations (there, 0..9). The formula probably is correct for real values, and should be close for floats. But the formula is not right for 8-bit values, and the difference is significant.

EXAMPLE

Suppose we have a total population of 4 values, 0..3: If the first sample is 0, there are 3 possibilities which continue a run up. If the first sample is 1, there are 2; if 2 then 1; and if the first sample is 3, there is no possibility of continuing a run up because there is no higher value in the population. So, out of the 16 possible combinations of values, only 6 are runs up of length 2 (*):

```
|      0 1 2 3
|   0  - * * *
|   1  - - * *
|   2  - - - *
|   3  - - - -
```

The figure shows all possible runs up of length 2: the first sample selects the row, and the next selects the column. The symbol - means we do not have a valid run up of 2; the symbol * means we do. So we find that the exact probability is 6 out of 16 instead of the 8 out of 16 we expect from the Knuth formula. This shows that the population size (the number of possible values) does indeed influence the probability value, independent of run length.

IMPLICATIONS

It is possible to count the number of acceptable byte combinations by scanning all possibilities of 2, 3, 4 bytes or more. Dividing the number of successes by the total number of possibilities gives the probability of success. Approaching the problem this way gives us a handful of results which are both unarguably correct and as precise as we can handle. For 8-bit values, pure brute force limits us to about 6 terms, which implies a maximum run length of only 5:

```
|    n   1/n!         by count
|
|    2   0.50000000   0.49804688
|    3   0.16666666   0.16471863
|    4   0.04166666   0.04069708
|    5   0.00833333   0.00801224
|    6   0.00138888   0.00130929
```

This generates a comparison by run length:

```
|    r    Knuth        counting      diff
|
|    1    0.500        0.502         0.002
|    2    0.333333     0.333328      0.000002
|    3    0.125        0.124         0.001
|    4    0.0333       0.0327        0.0006
|    5    0.0069       0.0067        0.0002
```

The difference between any pair of alternative probabilities is small. But this difference is multiplied by the number of samples. Then, in the chi-square computation, that result is squared and divided by the expected value. For a particular trial of 51200 samples (200 * 256), we would have the following expectations and results:

```
|    r   Knuth     counting  diff    sq      chi sq term diff
|
|    1   25600.0   25497.6   102.4   10486   0.4
|    2   17066.6   17066.4
|    3    6400.0    6348.8    51.2    2621   0.4
|    4    1705.0    1674.2    30.76    946   0.55
|    5     353.3     343.0    10.0     100   0.28
|                                            ----
|                                            1.63
```

So, over lengths 1..5, we see a chi-square bias of 1.63; including lengths 6 and 7 we might predict a total bias of about 2.0 (this does in fact occur at length 6). At 51,200 samples, we generally expect over 5 counts in each bin through r = 7, so we have 6 degrees of freedom. With a DF of 6, a chi-square difference of 2.0 is sufficient to move a probability from 0.90 to 0.95 or from 0.95 to 0.98, approximately, and from 0.01 to 0.05; this is a significant bias. This unbalance is obvious to the eye simply by noting the number of trials which have probabilities of 0.0xxxx versus those with probability 0.9xxxx.

EXPERIMENT

Exhaustive enumeration limits us to rather small term values, but this is hardly the only way to investigate the phenomena. Another way is to simply take a huge number of samples and report the accumulated probabilities. With 100M or 10**8 runs, we can hope to see 4 decimal digits of accuracy in the experimental values:

```
|    r   Experiment   Knuth        counting
|
|    1   0.50194788   0.50000000   0.50195393
|    2   0.33332273   0.33333333   0.33332825
|    3   0.12400923   0.12500000   0.12402155
|    4   0.03270330   0.03333333   0.03268484
|    5   0.00670438   0.00694444   0.00670295
|    6   0.00112878   0.00119048  -1.0
|    7   0.00016154   0.00017361  -1.0
|    8   0.00001972   0.00002205  -1.0
|    9   0.00000222   0.00000248  -1.0
|   10   0.00000021   0.00000025  -1.0
|   11   0.00000001   0.00000002  -1.0
```

* At r = 1 we see an exact 4-place match to counting results, with only a 2-place match for the equation.

* At r = 2 we get a 5-place match (versus only 4 for the equation), but do not have the accuracy to believe it, so this particular line is inconclusive.

* At r = 3 we see a 4-place match to counting, versus only 2 for the equation.

* At r = 4 we would have a 4-place match if we round to 4 places; the equation only matches 2.

* At r = 5 we again get a 5-place match to counting, versus only 3 for the equation.

* At r = 6 and above, there are no counting results.

Based on this data, one is forced to conclude that the experimental run length values match the counting values typically 100x better than the alternative. Further, this is as good as one could reasonably expect, given the size of the experiment.

THE COMBINATORIC MODEL

Consider what can happen with two samples from a population of 4:

```
|   S1  S2  run-up
|
|   0   0   -
|   0   1   *
|   0   2   *
|   0   3   *
|   1   0   -
|   1   1   -
|   1   2   *
|   1   3   *
|   2   0   -
|   2   1   -
|   2   2   -
|   2   3   *
|   3   0   -
|   3   1   -
|   3   2   -
|   3   3   -
|          ---
|           6
```

Here we have 6 ordered pairs which each can be a run up. We do not know that they are runs of length 2, because that depends upon having a run failure (a non-increasing value) in the *next* sample. But, clearly, each run of length 3 can only be built upon a successful run of length 2, so these pairs obviously constitute the only possibilities for all runs of length 2 or more.

The number of pairs which can participate in a run of length 2 is the number of symbols N taken 2 at a time, with no symbol repeated. This is just the number of *combinations* of the possible symbols taken 2 at a time. Each combination is a unique subset of the available symbols. Each of these combinations can be arranged either in an increasing or a decreasing order, so for each combination there is exactly one run up possibility and one run down possibility. Clearly, the number of possible runs up equals the number of possible runs down. We also note that N possibilities can be neither. So for length 2 we have:

```
|    N     N               N            2
|   ( ) + ( )  +  N  =  2 ( )  +  N  =  N
|    2     2               2
|
|   (runs up + runs down + neither = total possibilities)
```

so that

```
|    N        2
|   ( )  =  (N  - N) / 2  =  N (N-1) / 2
|    2
```

which is both correct and familiar.

For higher runs lengths, there are more possible orderings for each combination, but still only one of these will be a valid run up, and only one other will be a valid run down. So if we are just counting possibilities, we can ignore the idea of ordering within each combination. Since each possible combination will have a single success, we only need to find the number of combinations, which is a standard calculation.

COMPUTATION

The term probability function (the number of possibilities which can be part of a run up or run down divided by the total number of possibilities) is just another view of an old friend:

```
|               N      r
|   Pt(N,r) = ( ) / N
|               r
```

The number of combinations of population N taken r at a time is the number of possible runs up of length r *and higher*. So, to find the probability of runs of a particular length we have:

```
|   P(N,r) = Pt(N,r) - Pt(N,r+1)
```

which, for N = 256, produces the following correct results:

```
| Byte Runs Up/Dn Probability by Run Length
|
|    r   prob
|
|    1   0.501953125000
|    2   0.333328247070
|    3   0.124021545053
|    4   0.032684844686
|    5   0.006702946664
|    6   0.001126633669
|    7   0.000160449945
|    8   0.000019817478
|    9   0.000002159795
|   10   0.000000210491
|   11   0.000000018541
|   12   0.000000001489
|   13   0.000000000110
|   14   0.000000000007
|   15   0.000000000000
```

These values closely match the experimental results from 2 billion (2 * 10**9) total runs. (The first 5 lines each match the experimental results in 5 places after the decimal, lines 6, 7 and 9 match 6 places, line 8 matches 7 places, and line 10 matches 8 places.)

SUMMARY

A source of experimental bias in "runs up" RNG testing has been identified. This bias is a result of using a formula perhaps intended for real values in tests of small populations. Based on exact exhaustive results, the number of possible runs of length r and above corresponds to the number of possible combinations of N things taken r at a time, for population N. This turns out to be a very straightforward and believable combinatoric model for these tests. The computation also corresponds to experimentally determined values across a wide range of run lengths in massive experiments. Correct expectation probabilities for byte value runs up or down have been tabulated.

REFERENCE

1. Knuth, D. 1981. The Art of Computer Programming. Vol. II, Seminumerical Algorithms. Addison-Wesley.

[Terry Ritter](#)   [ritter@io.com](#)   [http://www.io.com/~ritter/](#)

[Terry Ritter](#), his [current address](#), and his [top page](#).

Last updated: 1997-12-25

Path: cisunx!ucsd!ucsc!news2!sparkler.net!news.sprintlink
.net!hookup!yoshua.marcam.com!Mathews.Cryptography.org.gsfnet.com
Newsgroups: sci.crypt,sci.math,sci.math.num-ana
From: ritter@ODI.com (Terry Ritter)
Newsgroups: sci.crypt
Subject: Fenced DES Block Cipher(via email gateway)
Date: 20 Apr 1994 11:27 -0600
Organization: Ethash Mail-to-News Gateway
Lines: 1000
Sender: cmendes@ix.netcom.com
Message-ID: <19940420120304.DA00047@Cellini3.SD.CONP-
NOTP.Pooling.Net> (ex.stream.ada)

Ritter Software Engineering
Fremont Center Office
Austin, Texas 78704
(512) 892-0494, ritter@ix.com

Fenced DES

Terry Ritter
April 17, 1994

Introduction

This article is one in a series which document my attempts to find a cipher which is cryptanalytically strong, in the Data Encryption Standard (DES), which I believe is now obsolete.

The intent is to find a relatively-simple and believable construct which uses DES as a building block, thus avoiding the need to certify a complete new cipher. I note that currently there is no construction which could and also would provide such certification.

In this article I propose a new "fenced" ciphering construct which uses DES as a building block. Fenced ciphering is a variation on tabes about tables (the computation tper Aytre) of DES alone, and may have the strength of four DES keys.

In this design, some important block cipher properties seem to follow logically from the widely-accepted existence of those properties in DES itself.

Wide Blocks

All practical block ciphers attempt to emulate a large substitution table. Unfortunately, the computation required for substitution seems to behave like a substitution table of 2**64 elements. Accepting DES as a reasonable design means that we have implicitly accepted the idea that 64-bit wide blocks are sufficiently large to avoid block cryptanalysis. Certainly, if a small-block substitution were acceptable, we would all use that simple and fast alternative instead of DES. Unfortunately, we do not have a proof that 64-bit wide block size is a significant factor in block cipher strength.

DES is often used to encipher language text, which contains a surprisingly small amount of information. Since most cryptanalytic programs routinely compress language text by 60%, we can expect that 64-bit blocks will contain at least 24 bits of language text, or of information, while it is not currently known how this could be exploited, a 256-bit-wide block should contain four times that much information, which should allow new cryptanalytic attacks.

A large block size also addresses some aspects of cryptanalytic "parallel" attacks. Such attacks may be viewed as a "divide-and-conquer" approach to attacking each plot from a large number of ciphertexts. With a 64-bit block about 2**32 ciphertext blocks would be expected to have the same plaintext value, while with a 256-bit block, 2**128 blocks using "defined plaintext," or 2**64 blocks using "chosen ciphertext blocks, which is completely out of the question. Thus, a 256-bit block would seem to be much more secure.

A large-block-size cipher need not expand ciphertext beyond the normal expansion for DES CBC Initialization vector and key-length block, provided that one trailing IV and one trailing key block can be used if needed. All the preceding blocks would be wi blocks.

The Two Problems

This project has had to address two major problems:

1. Weaknesses of Multi-Layer Constructs: Many simple multi-layer ciphers have been found to be weak, and it is easy to be working simultaneously on both the input and output layers, given "known plaintext" or "defined plaintext." In general, the "strength" of such constructs seems to be equivalent to one eight expect. This leads to three constructs which have "triple-DES" which tend to be very slow.

2. Weakness in Multi-Block Constructs: Similarly, simple large-block structures based on DES can be attacked by "parallel" attacks, which can be viewed as "divide-and-conquer" attacks on a 256-bit block cipher. Thus, a simple composite structure which does not have block bit effect every DES ciphering will have this weakness.

To expand the effective block size while using DES itself, fenced-DES ciphering uses a "block mixing transform" which is described in the previous article. In this article I will show how fully mixing those transforms can be used to create a cipher with a large block size, and with four keys, despite the mixing having no strength of its own.

The Block Mixing Transform

In a previous article I introduced the concept of a "block mixing transform" expanded from work by Ellis Bibbero as a tool to mix the information in two data blocks, and then recover that information. This concept could be expressed as two pairs of expressions:

```
X = f1( A, B );   Y = f2( A, B );

A = f3( X, Y );   B = f4( X, B );
```

The term "transform" is taken from the ability to change the values of a different data-space, and the ability to change the input values, and also the similarity to a "Fourier transform." The transform is "fully mixing" because the output bits of each block can be distinguished from the "mixing transformation" described by Shannon [28: 711].

The particular form I suggested was:

```
X = IA + B;   Y = IA + 2B;

A = IX + IY;   B = IX + 2Y;
```

with operations mod 2 and mod p, where p is some primitive mod 2 polynomial. The "mixing" operation was "fully mixing" in the sense that each bit of each block depended on every bit of both blocks. Such a transform is cryptanalytically secure [: This transform is a self-inverse, and thus is its own inverse. Thus, the same hardware can be used, and then a processing cost which is linear with block size.

Efficient implementation suggests a re-labeling as follows:

```
X = IA + 2B;   Y = IA + 3B;

A = IX + 2Y;   B = IX + 3Y;
```

Comments on the original "block mixing transform" article have uncovered a few other references to fixed-key such transforms, including Agarwal and Burrus [1], Pollard [9], Schnorr [10], and Ellis Bibbero. Unfortunately, this seems to be glad to hear of any other references to fixed-key such transforms.

The mixing transform need not be a cipher by itself. Indeed, it need have no "strength" at all, but must provide at least a "mixing" of the two blocks. Any "strong" cryptanalytic attack should also be unpalatable and fast. Although size is not an issue in most individual ciphering, speed is a major issue for block mixing, which must be applied to the entire data stream. The application in this article mixes blocks of substantial size, making many other forms of mixing completely impractical.

4x Fenced-DES

Consider the following construct:

```
 S S S S S S S S S S S S S S S S S S S S S S S S S S S S S S S S
 ---------------------------------------------------------------
 ---DES---------DES---------MIX---------DES---------DES---------
 ---------------MIX-------------------------------MIX-----------
 S S S S S S S S S S S S S S S S S S S S S S S S S S S S S S S S
```

Here each "S" represents an 8-bit substitution table. Thus, we have 32 independent substitution and 32 output substitutions, each a separately-shuffled and independent table, and an overall block size of 256 bits. Each substitution table is independently keyed. Each of three mixing and two DES stages is independently keyed. Note that the invertent mixing levels combine two 128-bit blocks, a "parallel" attack which is not nevertheless practical using the selected block cipher transform.

The idea is to spread the effect of each input bit to each of the 256 output bits, and to produce a particular output bit from a combination of all four DES results. If this works, the Opponent must attack each DES result in parallel, perhaps simultaneously to match a particular known-plaintext pair.

An experimental implementation of the above construct performs all 64 substitutions and all 8 mixings in take-time time, using 256 processors. Currently, it ciphers 8 times the data with about 8 times the delay of a single-block-size cipher using 32 DES S-Tables or so. Clearly higher implementation might be the DES computations in hardware.]

In the experimental implementation, table and key initialization take about 200 times the computation of a ciphering of a block. Key-dependent tables are a consequence of shuffling DES-shell substitution tables.] Even so, it is probably faster to compute the 16K initial tables than to initialize the Blowfish cipher with 4168 bytes. Or to expand DES. Construction is faster than expansion.

The keyed construction of the substitution tables implies the presence of a specific cryptanalytic ME. This means that any overall Fenced-DES specification will pin-down the key-dependent tables, and thus the ME, for all of the DES operations. The current implementation uses a fixed set of DES S-Tables and the nonlinear "jitterizer" [8] which I have discussed many times in the context of "Dynamic Substitution" and my other dynamic substitution DES ciphers.

In the experimental implementation, a user key of arbitrary length and content is hashed (CRC'ed) by 32 separate and independent primitive-mod-2 polynomials (11: through 16-nonatic), producing 32 32-bit values which are used to initialize the jitterizer [8] keys. Note that this approach eliminates the need for keys to have a specific format unique to this particular cipher. This approach allows the use of the same key-hashing for many different ciphers, all of which can use the same key format.

Deciphering simply uses inverse substitutions (the inverse of each encipher output substitution is used for decipher input) and DES deciphering. The jitter-based block mixing transform is a self-inverse and needs no changes.

Mixing Levels

The arrangement of the mixing levels deserves some comment. First, note that a change in any one input data bit produces a distribution of changes out of the associated input substitution, which affects the DES input block in 8 bits in two adjacent bytes and changes one possible byte input has a 50 percent probability of affecting each of the eight output bits from that substitution.

A substitution table is a one-to-one mapping of 8-bit values or output codes. An invertable substitution table with 256 entries, 0 to 255, has the property that if the output code j in a to, output S[i] j = i. This implies that S contains n different output codes.

An invertable substitution table S contains each output code value exactly once. Since each possible index selects a different output code, the output codes which differ in one input bit output code. Since different code values must differ in at least one bit, any input change must produce a change in at least one output bit.

Given invertable substitution table S with shuffled contents, the output change which will be caused by a change in input bit i is an arbitrary selection from the output codes which differ from the current output code. If the output codes are a complete random set, the output bit will change with probability 1/2. Since when that is is likely that about half of the output bits will change for any input code change of any nature whatsoever.

Conversely, since each output bit is produced by an output code, and the selected output code is completely dependent on input bit i, any input change which changes the output code on every bit of the input. A network with this property is normally called "complete" [5], and localized completeness is a well-known ME for "avalanche" [7 :72] in an iterated block cipher.

Mixing levels

The arrangement of the mixing levels deserves some comment. First, note that a change in any one input data bit produces a distribution of changes out of the associated input substitution, affecting the DES input block in 8 bits in two adjacent bytes and changes one possible byte input has a 50 percent probability of affecting each of the eight output bits from that substitution.

A substitution table is a one-to-one mapping of 8-bit values or output codes. An invertable substitution table with 256 entries, 0 to 255, has the property that if the output code j is in S, output S[i] j = i. This implies that S contains n different output codes.

An invertable substitution table S contains each output code value exactly once. Since each possible index selects a different output code, the output codes which differ in one input bit output code. Since different code values must differ in at least one bit, any input change must produce a change in at least one output bit.

Given invertable substitution table S with shuffled contents, the output change which will be caused by a change in input bit i is an arbitrary selection from the output codes which differ from the current output code. If the output codes are a complete random set, the output bit will change with probability 1/2. Since when that is is likely that about half of the output bits will change for any input code change of any nature whatsoever.

Conversely, since each output bit is produced by an output code, and the selected output code is completely dependent input bit i, any input change which changes the output code on every bit of the input. A network with this property is normally called "complete" [5], and localized completeness is a well-known ME for "avalanche" [7 :72] in an iterated block cipher.

Next, note that we first mix two 64-bit blocks (twice), then two 128-bit blocks. Suppose we have a change in one input data bit. This will change the output of one DES S-Table, and thus normally affect just a single DES block. But the 64-bit mixing extends those changes to two DES blocks, and the 128-bit mixing extends the change to four DES blocks. Thus, a single input change will affect four bits which affect all four DES operations.

Using the transformation X = IA + 2B; Y = IA + 3B; any value change in A or B must be reflected in both X and Y.

Suppose some change C is added to A:

```
A  := IA + 2B  (mod 2, mod p)
Y' := IA+C + 3B
   := Y + IC
dY := Y - Y' = IC
```

but IC is non-zero (thus affecting the output) for any C which is not zero, and if C is zero, then B must then be change to A.

Suppose some change C added to B:

```
X  := IA + 2B  (mod 2, mod p)
X' := IA + 2B+C
   := X + 2B + 2C
dX := X' - X = 2C
```

Similarly, 2C is also non-zero for any C which is not zero.

Suppose we try to make C half the value of p plus the highest bit (2**(deg(p)-1)) so that p will be added to 2C to cancel the lower bits of p: Alas, p is irreducible so there is no X [ ] to 2C = p

Similar arguments apply for Y = IA + 3B.

The experimental implementation uses the degree-128 irreducible 0100000000000000000000000000000001 (hex), and the degree-64 irreducible 0000000000000001 as block mixing polynomials.

The output from each DES operation is, of course, random-time, so one might think it could be used directly. Unfortunately, this time level structure is still necessary to prevent, for example, "fix-up" attacks which would be possible if the DES operations did not all affect just a single DES block. But the 64-bit mixing extends those changes to two DES blocks, and the 128-bit mixing extends the change to four DES blocks. Thus, a single input change will affect four bits which affect all four DES operations.

Suppose that the "strength" for a Fenced-DES is (1/2**56)*(1/2**64)*(1/2**64), a very big number. I would be obligated to learn of a simpler attack.

It would of course be ridiculous to accept this sort of number as "strength" for a cipher which is based on DES, which is not anything such bits, since this should be sufficient for the next couple of decades and then we may have a stronger basis for cryptographic design.

Design Strength

Note that we need assume no "strength" for the mixing layers, but only that they be fully mixed. Each mixed output block must be a function of each and every bit in both input blocks. In this particular design we need only two levels of mixing to make sure that every input bit has influence over all 256 output bits. Three levels of mixing is to make sure that four DES blocks participate in every output bit.

The purpose of the small substitutions is to prevent the (weak and simple) mixing from being exploited by simple cryptanalytic attacks. Fenced-DES uses fully-shuffled substitutions which are intended to isolate the mixing functions, because "known plaintext" is only equal to the input of the first substitution if the first substitution is equal to the output. When known plaintext is not available, and substitutions cannot be separated for divide-and-conquer, little substitutions can be surprisingly strong.

In the 4x construct, we might lay all the strength on the four DES operations, and use the mixing only to prevent "divide-and-conquer" attacks which are able to isolate one of the DES keys (perhaps as a consequence of its operation using the same state), would reduce the DES key to 56 bits. Thus the substitutions must be keyed even if we choose their "strength."

Strength Supported by Attack

Exhaustive Search: try each key until the correct one is found. Preventing this may requires a keyspace substantially larger than 56 bits (or, with a computationally-expensive setup phase, perhaps 56 bits). Fenced-DES substitutions are keyed by a hashed user key at least 128 keyspace. Note that this is not four times the DES keyspace, but four times the key size, which is 2**128 times the conventional DES keyspace.

Known-Plaintext/Defined Plaintext: Somehow "obtain" both the plaintext and the ciphertext, and then search for the correct number of enciphering under one key. This has many flaws.

Codebook: Try to obtain all possible ciphertexts and associated plaintext. Then, when a ciphertext occurs, look it up. This is normally prevented by having a large number of transformations, each keyed by a large block size and a large keyspace. Fenced-DES has this.

Codebook approaches can be combined with "divide-and-conquer" to isolate and define parts of some ciphers. Fenced-DES tries to avoid these attacks by not allowing the parts to be isolated and worked on separately.

Meet-in-the-Middle: With a multi-layered structure, given known-plaintext and the ciphertext, try all possible first-layer possible result, and search the bottom keyspace to find every possible value. With a two-level construct, matches can be found with the sum of the keyspaces rather the product. Fenced-DES avoids this by using a three-level construction, and by using outer layers which have a huge "keyspace."

Differential Cryptanalysis: Given a 16-round Feistel cipher with known tables, use any statistical unbalance in the tables to peer into the cipher and determine the key. Fenced-DES uses fixed tables, by using only balanced full-substitution tables, and by using a fully-balanced block mixing transform to avoid "divide-and-conquer."

Important Aspects of the Design

First, the Fenced-DES construct is more like a Kam-Davida construct than a Feistel cipher. Unlike a Feistel cipher, an iterated Feistel design [5] represented by DES block mixing mixing transform is specifically intended to avoid the sort of weakness exploited by the recent Meet-Tweaners attack [4] on DES designs.

Next, it seems that there is a fundamental weakness in any two-layer cipher, given "known plaintext" or "defined plaintext" and assume "defined plaintext" capabilities. Fenced-DES uses three independent layers to avoid such attacks.

Conventional block-cipher designs generally use unkeyed static substitution tables which are selected for "optimal" performance. Such tables are then used in a fixed-key cipher which is used as any table-generation is as good as any other, making substitution unnecessary. (A shuffled substitution is very unlikely to be "optimal" in any sense.) Such fixed-table substitutions can be detected easily. The mid-level substitution-here DES--acts to hide any S-box linearity.]

Conventional block-cipher designs are also very economical with state, using either small tables (e.g. the 256 bytes in eight S-boxes in Blowfish) or a single table (e.g. the 256 bytes in DES). But no Fenced-DES uses 16K bytes of tables, all keyed.

More conventional S-F designs tend to use the same block size at each substitution level, thus becoming vulnerable to "divide-and-conquer" attacks [4]. Fenced-DES differs from this approach by having a block size which is twice the block size of the substitution layers (this is similar to a Kam-Davida "particle" [1: 74] but differs in that it is a single block). This should prevent these attacks on substitution-layer-based ciphers, because the blocks are being separated and attacked individually.

Other contemporary block-cipher designs generally use a 64-bit block size. This is much weaker than it was 20 years ago, when that size was selected for DES. To avoid birthday attacks on ciphertext, we must limit the ciphertext quantity to about 2**32 Fenced-DES has a reduced block size (256 bits) and uses three independent layers to avoid such attacks.

Conventional block-cipher designs generally use unkeyed static substitution tables which are selected for "optimal" performance. Such tables are then used in a fixed-key cipher which is used as any table-generation is as good as any other, making substitution unnecessary. (A shuffled substitution is very unlikely to be "optimal" in any sense.) Such fixed-table substitutions can be detected easily. The mid-level substitution-here DES--acts to hide any S-box linearity.]

Conventional block-cipher designs are also very economical with state, using either small tables (e.g. the 256 bytes in eight S-boxes in Blowfish) or a single table (e.g. the 256 bytes in DES). But no Fenced-DES uses 16K bytes of tables, all keyed.

More conventional S-F designs tend to use the same block size at each substitution level, thus becoming vulnerable to "divide-and-conquer" attacks [4]. Fenced-DES differs from this approach by having a block size which is twice the block size of the substitution layers (this is similar to a Kam-Davida "particle" [1: 74] but differs in that it is a single block). This should prevent these attacks on substitution-layer-based ciphers, because the blocks are being separated and attacked individually.

Other contemporary block-cipher designs generally use a 64-bit block size. This is much weaker than it was 20 years ago, when that size was selected for DES. To avoid birthday attacks on ciphertext, we must limit the ciphertext quantity to about 2**32 Fenced-DES has a reduced block size (256 bits) and uses three independent layers to avoid such attacks.

A fundamental difference is that conventional S-F designs perform a single block cipher operation on the entire plaintext block. Fenced-DES uses a substantial I/O operation and processes exactly the same as some output bit in the precious layer. Conventional S-F designs use a single block cipher operation of a block mixing transform to make the input code to a double-layer substitution (in this case, DES) a function of every substitution in the previous layer. Fenced-DES uses block mixing to make an expanded while preventing substitutions on the tables from being separated and attacked individually.

An interesting aspect of the Fenced-DES design is the possibility that assumed properties of DES--a cipher which has been studied and used for almost 20 years--can be provably expanded into properties of the larger cipher.

Summary

A new type of cryptographic ciphering construct has been introduced which uses DES as a building block. The result seems to provide a larger block size and more strength than triple-DES (the leading alternative), while operating almost three times as fast.

References

[1] Agarwal, R. and C. Burrus. 1974. Fast Convolution Using Fermat Number Transforms with Applications to Digital Filtering. IEEE Transactions on Acoustics, Speech, and Signal Processing. ASSP-22(2): 87-97.

[2] Ayoub, F. 1982. Probabilistic completeness of substitution-permutation encryption. IEE Proceedings, Pt. E. 129(5): 195-196.

[3] Feistel, H. 1973. Cryptography and Computer Privacy. Scientific American. 228(5): 15-23.

[4] Meyn, M. and D. Teuenie. 1993. Cryptanalysis of Tree-Structured Substitution-Permutation Networks. Electronics Letters. 29(11): 40-41.

[5] Kam, J. and G. Davida. 1979. Structured Design of Substitution-Permutation Encryption Networks. IEEE Transactions on Computers. C-28(10): 747-753.

[6] Pollard, J. M. 1971. The Fast Fourier Transform in a Finite Field. Mathematics of Computation. 25(114): 365-374.

[7] Rueppel, R. 1992. Stateless Convolution via Recursive Transforms. IEEE Transactions on Computers. C-21(12): 1268-1272.

[8] Ritter, T. 1991. The Efficient Generation of Cryptographic Confusion Sequences. Cryptologia. 15(3): 81-339.

[9] Ritter, T. 1990. Substitution Cipher with Pseudo-Random Shuffling: The Dynamic Substitution Combiner. Cryptologia. 14(4): 274-302.

[10] Shannon, C. 1949. Communication Theory of Secrecy Systems. Bell System Technical Journal. 28: 656-715.

Terry Ritter ritter@ix.com

Generated on Wed Apr 20 09:00:00 2011 UTC (24 hours after GMT)
Powered by MediaWiki, Beautiful Soup, and PyPeg

Horst Feistel
IBM Thomas J. Watson Research Center
Yorktown Heights, New York
Walter A. Notz
IBM Research Center Zurich (Temp.)
Zurich, Switzerland
J. Lynn Smith, Jr. Data Security
Lexington, Massachusetts
Mitchel Kelsey, Cryptanalytic Research
Message Dr. Communications Research Division
Fort Meade, Maryland

Cryptography and Computer Privacy

Horst Feistel
Walter A. Notz
J. Lynn Smith

The Center of the Present Ad Hoc Group

Terry Brown
Carl Meyer

This article describes the Feistel-DES cipher and its theory of design. Feistel and Coppersmith have developed a family of ciphers based on the Feistel-DES cipher, and have developed a theory of design that has not yet been published in full.

The Feistel-DES cipher is a product block cipher design so that it does not have the weaknesses of substitution-permutation networks. Feistel-DES is a product cipher because it is composed of a sequence of "permutation" or reversal substitutions. Because Feistel-DES is a product cipher, it is possible to analyze the cipher by a theory of linear and differential cryptanalysis in a manner similar to that of Feistel-DES.

The Feistel-DES is the first I know to upgrade the strength of DES by a factor of 2 in the number of operations required to break the cipher. Feistel-DES is a product cipher that is composed of a sequence of reversal substitutions. Because Feistel-DES is a product cipher, it is possible to analyze the cipher by a theory of linear and differential cryptanalysis.

Table of Contents

1. Introduction
2. Cryptanalytic Approaches to Feistel-DES
2.1 Linear Attack
2.2 Differential Attack
2.3 The Block Cipher Complexity
2.4 Cryptanalytic Complexity
3. Feistel-DES
3.1 Substitution-Permutation Networks
3.2 Feistel-DES Substitutions
3.3 Cryptanalysis of Large-Block Feistel-DES
3.4 Feistel-DES Design
4. Block-Cipher Cryptanalysis
4.1 Linear Cryptanalysis of Feistel-DES
4.2 Differential Cryptanalysis
4.3 Feistel-DES
4.4 Cryptanalysis
5. Feistel-DES Design
5.1 Cryptanalytic Complexity
5.2 Feistel-DES Design
5.3 Cryptanalysis of Feistel-DES with Linear Substitutions
5.4 Cryptanalysis of Feistel-DES with Linear Substitutions
5.5 Strength of a Feistel-DES
5.6 Cryptanalysis of Feistel-DES
5.7 Strength of a Feistel-DES
5.8 Conclusion
6. Feistel-DES Design
6.1 Conclusion
6.2 Cryptanalysis
6.3 Differential Cryptanalysis
7. Conclusion
8. References

1. Introduction

The Data Encryption Standard (DES) has remained unchanged for almost twenty years. During that period the cryptanalytic attacks on DES have been mainly theoretical. The best attack against DES is still a brute force attack. With the same attacks can be applied to Feistel-DES, the cryptanalytic attacks on Feistel-DES are not theoretical. The best attack against Feistel-DES is a brute force attack. Because Feistel-DES is a product cipher, it is possible to analyze the cipher by a theory of linear and differential cryptanalysis in a manner similar to that of Feistel-DES. Because Feistel-DES is a product cipher, it is possible to analyze the cipher by a theory of linear and differential cryptanalysis in a manner similar to that of Feistel-DES.

2. Cryptanalytic Approaches to Feistel-DES

2.1 Linear Attack

The Feistel-DES cipher is a product cipher so that the linear cryptanalytic attacks have been mainly theoretical. The best attack against Feistel-DES is still a brute force attack. The Feistel-DES cipher is a product cipher so that the linear cryptanalytic attacks have been mainly theoretical. The best attack against Feistel-DES is still a brute force attack. The Feistel-DES cipher is a product cipher so that the linear cryptanalytic attacks have been mainly theoretical. The best attack against Feistel-DES is still a brute force attack.

2.2 Differential Attack

The Feistel-DES cipher is a product cipher so that the differential cryptanalytic attacks have been mainly theoretical. The best attack against Feistel-DES is still a brute force attack. The Feistel-DES cipher is a product cipher so that the differential cryptanalytic attacks have been mainly theoretical. The best attack against Feistel-DES is still a brute force attack. The Feistel-DES cipher is a product cipher so that the differential cryptanalytic attacks have been mainly theoretical. The best attack against Feistel-DES is still a brute force attack.

2.3 The Block Cipher Complexity

The Feistel-DES cipher is a product cipher so that the block cipher complexity has been mainly theoretical. The best attack against Feistel-DES is still a brute force attack. The Feistel-DES cipher is a product cipher so that the block cipher complexity has been mainly theoretical. The best attack against Feistel-DES is still a brute force attack. The Feistel-DES cipher is a product cipher so that the block cipher complexity has been mainly theoretical. The best attack against Feistel-DES is still a brute force attack.

2.4 Cryptanalytic Complexity

The Feistel-DES cipher is a product cipher so that the cryptanalytic complexity has been mainly theoretical. The best attack against Feistel-DES is still a brute force attack. The Feistel-DES cipher is a product cipher so that the cryptanalytic complexity has been mainly theoretical. The best attack against Feistel-DES is still a brute force attack. The Feistel-DES cipher is a product cipher so that the cryptanalytic complexity has been mainly theoretical. The best attack against Feistel-DES is still a brute force attack.

3. Feistel-DES

3.1 Substitution-Permutation Networks

The Feistel-DES cipher is a product cipher so that the substitution-permutation networks have been mainly theoretical. The best attack against Feistel-DES is still a brute force attack. The Feistel-DES cipher is a product cipher so that the substitution-permutation networks have been mainly theoretical. The best attack against Feistel-DES is still a brute force attack. The Feistel-DES cipher is a product cipher so that the substitution-permutation networks have been mainly theoretical. The best attack against Feistel-DES is still a brute force attack.

3.2 Feistel-DES Substitutions

The Feistel-DES cipher is a product cipher so that the Feistel-DES substitutions have been mainly theoretical. The best attack against Feistel-DES is still a brute force attack. The Feistel-DES cipher is a product cipher so that the Feistel-DES substitutions have been mainly theoretical. The best attack against Feistel-DES is still a brute force attack. The Feistel-DES cipher is a product cipher so that the Feistel-DES substitutions have been mainly theoretical. The best attack against Feistel-DES is still a brute force attack.

3.3 Cryptanalysis of Large-Block Feistel-DES

The Feistel-DES cipher is a product cipher so that the cryptanalysis of large-block Feistel-DES has been mainly theoretical. The best attack against Feistel-DES is still a brute force attack. The Feistel-DES cipher is a product cipher so that the cryptanalysis of large-block Feistel-DES has been mainly theoretical. The best attack against Feistel-DES is still a brute force attack. The Feistel-DES cipher is a product cipher so that the cryptanalysis of large-block Feistel-DES has been mainly theoretical. The best attack against Feistel-DES is still a brute force attack.

3.4 Feistel-DES Design

The Feistel-DES cipher is a product cipher so that the Feistel-DES design has been mainly theoretical. The best attack against Feistel-DES is still a brute force attack. The Feistel-DES cipher is a product cipher so that the Feistel-DES design has been mainly theoretical. The best attack against Feistel-DES is still a brute force attack. The Feistel-DES cipher is a product cipher so that the Feistel-DES design has been mainly theoretical. The best attack against Feistel-DES is still a brute force attack.

4. Block-Cipher Cryptanalysis

4.1 Linear Cryptanalysis of Feistel-DES

The Feistel-DES cipher is a product cipher so that the linear cryptanalysis of Feistel-DES has been mainly theoretical. The best attack against Feistel-DES is still a brute force attack. The Feistel-DES cipher is a product cipher so that the linear cryptanalysis of Feistel-DES has been mainly theoretical. The best attack against Feistel-DES is still a brute force attack. The Feistel-DES cipher is a product cipher so that the linear cryptanalysis of Feistel-DES has been mainly theoretical. The best attack against Feistel-DES is still a brute force attack.

4.2 Differential Cryptanalysis

The Feistel-DES cipher is a product cipher so that the differential cryptanalysis of Feistel-DES has been mainly theoretical. The best attack against Feistel-DES is still a brute force attack. The Feistel-DES cipher is a product cipher so that the differential cryptanalysis of Feistel-DES has been mainly theoretical. The best attack against Feistel-DES is still a brute force attack. The Feistel-DES cipher is a product cipher so that the differential cryptanalysis of Feistel-DES has been mainly theoretical. The best attack against Feistel-DES is still a brute force attack.

4.3 Feistel-DES

The Feistel-DES cipher is a product cipher so that the Feistel-DES has been mainly theoretical. The best attack against Feistel-DES is still a brute force attack. The Feistel-DES cipher is a product cipher so that the Feistel-DES has been mainly theoretical. The best attack against Feistel-DES is still a brute force attack. The Feistel-DES cipher is a product cipher so that the Feistel-DES has been mainly theoretical. The best attack against Feistel-DES is still a brute force attack.

4.4 Cryptanalysis

The Feistel-DES cipher is a product cipher so that the cryptanalysis has been mainly theoretical. The best attack against Feistel-DES is still a brute force attack. The Feistel-DES cipher is a product cipher so that the cryptanalysis has been mainly theoretical. The best attack against Feistel-DES is still a brute force attack. The Feistel-DES cipher is a product cipher so that the cryptanalysis has been mainly theoretical. The best attack against Feistel-DES is still a brute force attack.

5. Feistel-DES Design

5.1 Cryptanalytic Complexity

The Feistel-DES cipher is a product cipher so that the cryptanalytic complexity has been mainly theoretical. The best attack against Feistel-DES is still a brute force attack. The Feistel-DES cipher is a product cipher so that the cryptanalytic complexity has been mainly theoretical. The best attack against Feistel-DES is still a brute force attack. The Feistel-DES cipher is a product cipher so that the cryptanalytic complexity has been mainly theoretical. The best attack against Feistel-DES is still a brute force attack.

5.2 Feistel-DES Design

The Feistel-DES cipher is a product cipher so that the Feistel-DES design has been mainly theoretical. The best attack against Feistel-DES is still a brute force attack. The Feistel-DES cipher is a product cipher so that the Feistel-DES design has been mainly theoretical. The best attack against Feistel-DES is still a brute force attack. The Feistel-DES cipher is a product cipher so that the Feistel-DES design has been mainly theoretical. The best attack against Feistel-DES is still a brute force attack.

5.3 Cryptanalysis of Feistel-DES with Linear Substitutions

The Feistel-DES cipher is a product cipher so that the cryptanalysis of Feistel-DES with linear substitutions has been mainly theoretical. The best attack against Feistel-DES is still a brute force attack. The Feistel-DES cipher is a product cipher so that the cryptanalysis of Feistel-DES with linear substitutions has been mainly theoretical. The best attack against Feistel-DES is still a brute force attack. The Feistel-DES cipher is a product cipher so that the cryptanalysis of Feistel-DES with linear substitutions has been mainly theoretical. The best attack against Feistel-DES is still a brute force attack.

5.4 Cryptanalysis of Feistel-DES with Linear Substitutions

The Feistel-DES cipher is a product cipher so that the cryptanalysis of Feistel-DES with linear substitutions has been mainly theoretical. The best attack against Feistel-DES is still a brute force attack. The Feistel-DES cipher is a product cipher so that the cryptanalysis of Feistel-DES with linear substitutions has been mainly theoretical. The best attack against Feistel-DES is still a brute force attack. The Feistel-DES cipher is a product cipher so that the cryptanalysis of Feistel-DES with linear substitutions has been mainly theoretical. The best attack against Feistel-DES is still a brute force attack.

5.5 Strength of a Feistel-DES

The Feistel-DES cipher is a product cipher so that the strength of a Feistel-DES has been mainly theoretical. The best attack against Feistel-DES is still a brute force attack. The Feistel-DES cipher is a product cipher so that the strength of a Feistel-DES has been mainly theoretical. The best attack against Feistel-DES is still a brute force attack. The Feistel-DES cipher is a product cipher so that the strength of a Feistel-DES has been mainly theoretical. The best attack against Feistel-DES is still a brute force attack.

5.6 Cryptanalysis of Feistel-DES

The Feistel-DES cipher is a product cipher so that the cryptanalysis of Feistel-DES has been mainly theoretical. The best attack against Feistel-DES is still a brute force attack. The Feistel-DES cipher is a product cipher so that the cryptanalysis of Feistel-DES has been mainly theoretical. The best attack against Feistel-DES is still a brute force attack. The Feistel-DES cipher is a product cipher so that the cryptanalysis of Feistel-DES has been mainly theoretical. The best attack against Feistel-DES is still a brute force attack.

5.7 Strength of a Feistel-DES

The Feistel-DES cipher is a product cipher so that the strength of a Feistel-DES has been mainly theoretical. The best attack against Feistel-DES is still a brute force attack. The Feistel-DES cipher is a product cipher so that the strength of a Feistel-DES has been mainly theoretical. The best attack against Feistel-DES is still a brute force attack. The Feistel-DES cipher is a product cipher so that the strength of a Feistel-DES has been mainly theoretical. The best attack against Feistel-DES is still a brute force attack.

5.8 Conclusion

The Feistel-DES cipher is a product cipher so that the conclusion has been mainly theoretical. The best attack against Feistel-DES is still a brute force attack. The Feistel-DES cipher is a product cipher so that the conclusion has been mainly theoretical. The best attack against Feistel-DES is still a brute force attack. The Feistel-DES cipher is a product cipher so that the conclusion has been mainly theoretical. The best attack against Feistel-DES is still a brute force attack.

6. Feistel-DES Design

6.1 Conclusion

The Feistel-DES cipher is a product cipher so that the conclusion has been mainly theoretical. The best attack against Feistel-DES is still a brute force attack. The Feistel-DES cipher is a product cipher so that the conclusion has been mainly theoretical. The best attack against Feistel-DES is still a brute force attack. The Feistel-DES cipher is a product cipher so that the conclusion has been mainly theoretical. The best attack against Feistel-DES is still a brute force attack.

6.2 Cryptanalysis

The Feistel-DES cipher is a product cipher so that the cryptanalysis has been mainly theoretical. The best attack against Feistel-DES is still a brute force attack. The Feistel-DES cipher is a product cipher so that the cryptanalysis has been mainly theoretical. The best attack against Feistel-DES is still a brute force attack. The Feistel-DES cipher is a product cipher so that the cryptanalysis has been mainly theoretical. The best attack against Feistel-DES is still a brute force attack.

6.3 Differential Cryptanalysis

The Feistel-DES cipher is a product cipher so that the differential cryptanalysis has been mainly theoretical. The best attack against Feistel-DES is still a brute force attack. The Feistel-DES cipher is a product cipher so that the differential cryptanalysis has been mainly theoretical. The best attack against Feistel-DES is still a brute force attack. The Feistel-DES cipher is a product cipher so that the differential cryptanalysis has been mainly theoretical. The best attack against Feistel-DES is still a brute force attack.

7. Conclusion

In conclusion, we have shown that the Feistel-DES cipher, the Feistel-DES design, and the cryptanalysis of Feistel-DES have been mainly theoretical. The best attack against Feistel-DES is still a brute force attack. The Feistel-DES cipher is a product cipher so that the Feistel-DES design, the Feistel-DES cipher, and the cryptanalysis of Feistel-DES have been mainly theoretical. The best attack against Feistel-DES is still a brute force attack. The Feistel-DES cipher is a product cipher so that the Feistel-DES design, the Feistel-DES cipher, and the cryptanalysis of Feistel-DES have been mainly theoretical. The best attack against Feistel-DES is still a brute force attack. The Feistel-DES cipher is a product cipher so that the Feistel-DES design, the Feistel-DES cipher, and the cryptanalysis of Feistel-DES have been mainly theoretical. The best attack against Feistel-DES is still a brute force attack.

8. References

[1] Feistel, H., 1973. "Cryptography and Computer Security", Scientific American.

[2] Notz, W. 1981. "Differential Cryptanalysis of Feistel-DES", Proceedings of the Crypto 81 Workshop, Santa Barbara, CA.

[3] Feistel, H. and Notz, W. 1982. "DES - Differential Cryptanalysis", Proceedings of the Crypto 82 Workshop, Santa Barbara, CA.

[4] Brown, L., Kwan, M., Pieprzyk, J. and Seberry, J. "A Simple Substitution-Permutation Cryptanalysis of Feistel-DES", Proceedings of Crypto 88.

[5] Brown, L., 1988. A Proposed Design for an Extended DES, Proceedings of the Crypto 88 Workshop, Santa Barbara, CA.

[6] Feistel, H. and Coppersmith, D. 1989. "Block Cipher Design", Proceedings of the Crypto 89 Workshop, Santa Barbara, CA.

[1] Jakobsen, M. and K. Nyberg. 1995. On Exponent Sum of S-box Tables and Bounds on the Number of Differentials for Differential Cryptanalysis. Advances in Cryptology - ASIACRYPT'95. Springer-Verlag. 1163 of LNCS. 1-7.

[2] Coppersmith, D. 1993. Cryptanalysis and Computer Privacy. Advances in Cryptology - CRYPTO'93.

[3] Coppersmith, D., M. Matsui and S. Mister. 1995. DES Cryptanalysis: Linear Cryptanalysis Using Extended Linear Approximations. Proceedings of the DIMACS - Series 23 DES Cryptanalysis.

[4] Carlisle, C. and M. Cunningham. 1992. The Cryptanalysis of the Cryptographic Function Used in the Generation of the Computer Security Service. 4(1): 37-44.

[5] Daemen, J. and R. Govaerts. 1993. Cryptanalysis of Iterated Block Ciphers Using Collisions. Proceedings of the Workshop on Cryptanalysis.

[6] Dawson, C. and M. Goldberg. 1997. Randomized Block Ciphers Resistant Both Linear and Differential Cryptanalysis. Data Encryption Workshop.

[7] Feng, D. and W. Wuchen. 1995. Structural Analysis of Cryptographic Functions Based on Linear Approximation. Transactions on Computers. 1:(8):767-770.

[8] Feistel, H. 1973. A Cipher by Mail. Cryptobytes. July 1 (2).

[9] Knudsen, L. 1995. The Art of Cryptanalysis. Vol. 1. Advances in Cryptology - ASIACRYPT'95. Springer-Verlag. 1163 of LNCS. 148-167.

[10] Lai, X. and J. Massey. 1992. How to Construct Pseudorandom Functions from Pseudorandom Permutations. Journal of Cryptology. 1(1):11-24.

[11] Lai, X. and J. Massey. 1992. A Proposal for a New Block Encryption Standard. Advances in Cryptology - EUROCRYPT'92. Springer-Verlag. 1992 of LNCS. 389-404.

[12] Matsuzaka, K. and J. Tsuji. 1995. Matrices and the Structure of Ciphers. Advances in Cryptology - ASIACRYPT'95. Springer-Verlag. 1163 of LNCS.

[13] Merkle, R. and M. Hellman. 1981. On the Security of Multiple Encryption. Communications of the ACM. 24(7):465-467.

[14] Merkle, R. 1990. Fast Software Encryption Functions. Advances in Cryptology - CRYPTO'90.

[15] Nyberg, K. and L. Knudsen. 1995. Provable Security Against a Differential Attack. Advances in Cryptology - CRYPTO'95. Springer-Verlag. 1995 of LNCS. 566-574.

[16] Pieprzyk, J. and B. Sadeghiyan. 1995. Design of Substitution-Permutation Networks Resistant to Differential and Linear Cryptanalysis. ASI Proceedings. Part 2 of Lecture Notes in Computer Science.

[17] Pieprzyk, J. and B. Sadeghiyan. 1995. Methodology for Cryptanalysis of Substitution-Permutation Networks. Proceedings of the 1st Australian Conference on Information Security and Privacy. Springer-Verlag. 1172 of LNCS. 1-12.

[18] Pieprzyk, J. 1995. Error propagation property and application in cryptography. IEE Proceedings - Part E Computers and Digital Techniques.

[19] Rijmen, V. 1996. The Efficient Generation of Cryptographic Confusion Sequences. Proceedings of WCC'96.

[20] Schneier, B. 1995. Cryptography for Number Theorists. Proceedings, Berlin Workshop on Cryptology. Springer-Verlag. 1000 of LNCS.

[21] Tardy-Corfdir, A. 1995. Multiple Extension and Statistical Attacks on Block Ciphers. Proceedings of Eurocrypt'95.

[22] Vaudenay, S. and J. Tsuzuki. 1995. On the Security of S-Boxes. Proceedings of Eurocrypt'95. Springer-Verlag.

[23] Wagner, M. 1995. Efficient DES Key Search. (in press).

This document was created using Adobe Acrobat PDF Merge.

Huge Block Size Discussion

Contents

- Germano Caronni "current data encryption techniques usually encrypt a data stream or small blocks of data." ". . . [this] is perhaps not . . . the strongest way to secure data." "Imagine a data file . . . being encrypted as a whole, which means that each of the output bits depends on each of the input bits (and naturally the secret key)."
- George Barwood "I think that the computation cost would be proportional to at least the square of the block size . . ."
- Marc Thibault "That's not intuitively obvious. As the block size goes up it should be possible to reduce the complexity of the algorithm and still maintain strength."
- Bryan Olson "While block size should be large enough to foil exhaustive search over blocks, there's little evidence that it needs to be larger."
- John Michener "View the primitive block code, DES, as a very large S-Box. Encrypt the the data with DES to mix all the bits within consecutive 8 byte blocks. Transpose the data either bitwise, one bit per subsequent 64 bit block to expand the diffusion 64X, or 1 byte ber 8 byte block to expand the diffusion 8 X."
- Lloyd Miller "You just do two passes with a OFB type system. Do the second pass backwards."
- Donald T. Davis "before encrypting for the archive, prepend the file's message-digest to the plaintext. this will make every ciphertext bit depend on every plaintext bit."
- Dr. mike "One comment pointed out that the encryption time expands with the block size. I did look at this once (very briefly) with the idea of using blocks on the order of 32K or so using elliptic curve math. It can be done. It would take a depressingly long time to encrypt a single block, many minutes on a 400 MHz Alpha (as in >100)."
- W T Shaw "This depends on algorithm . . . ."
- Terry Ritter "I have been working on this for some time. I have developed Fenced DES and Variable Size Block Cipher technologies, both of which are patent pending . . . ."
- Jerry Leichter "Ritter's arguments are strongest when applied to file (or disk) encryption, and larger block sizes might be very appropriate there . . . ."
- W T Shaw "By keeping block size as small as is common, the strengths of the algorithms are crippled so as to make it difficult to make them increasingly stronger."
- Bryan Olson "I'd invite anyone who has closely examined Terry's 'Fenced DES' to comment on the attack I proposed in another thread of this same topic."
- Bryan Olson "Large-block ciphers seem to require more operations to achieve the same level of diffusion, and consequently most settle for less."
- Terry Ritter "Modern mixing guarantees distribution of any input change to every substitution in the next layer in n log n operations."
- Bryan Olson "So I have to search over a 2^32 space, and run a few hundred pairs for each to find such a match. Once I've found it, I have a lot of information about the four permutations which formed my four critical bytes. Repeating the process to find all the matches over all 32 permutions, I should be able to break the initial permutations."
- Terry Ritter "This attack indicates that any system using this Fenced DES design must prevent a chosen-plaintext attack of the necessary size."
- Robert Scott "As impractical as these attacks seem, the fact that they are theoretically possible is going to make a cipher unacceptable for adoption as a standard."
  [It turns out that this fact is in fact not. -- TFR]
- Terry Ritter "Actually, I agree . . . ."
- Robert Scott "You mentioned before that there may be value to using DES as a component because it has been so scrutinized and is so widely accepted. This may be a psycological plus, but the official status of DES does not impress me very much."
- Boudewijn W. Ch. Visser "Since todays cipher may very well end up in a high-bandwith application, I don't consider such requirements completely 'academically'." "A modern cipher should also be suitable to encrypt a few gigabytes of data with no keychange for months or years,and lots of known or even chosen plaintext."
- Bryan Olson "I can justify calling the attack successful cryptanalysis of 4X Fenced DES (provided I can advance it at a few points)."
  *[As it turned out, he could **not**! See Fenced DES Attack Discussion.]*
- Terry Ritter "I am probably the last person to judge such an attack on its overall qualities, because if even *part* of the attack can do something I did not think could be done, it has already triggered a redesign."
  [What was "triggered" was the avoidance of this sort of thing in new designs; Fenced DES is fine as originally proposed.]
- Lewis McCarthy "I believe it is reasonable to expect an implementor to decide upon a range or fixed number of rounds of a cipher (hash fn., etc.) when integrating cryptography into an application."
- Roger Fleming "I agree wholeheartedly that even for high entropy texts, 64 bit wide blocks will one day (real soon now) reach the end of their useful lives. But that doesn't mean we have to go all the way to 1600 bit blocks."
- Terry Ritter "from the design standpoint, once one has a flexible efficient, large-block technology, *why not* use large blocks and keys?"
- W T Shaw "Given the same algorithm, the larger the block, the more possibilities for keys, fewer blocks are used, and it is apt to beharder to attack by brute force."
- Terry Ritter "Certainly it is easier to build a chip with a 64 data I/O pins than one with 1600 I/O pins; therefore, don't do that."
- Bruce Schneier "I kind of like the idea, but I worry about its security."
- Wayne Schlitt "Is it really that important that _every_ plaintext bit gets diffused to _every_ ciphertext bit?" ". . . it seams reasonable to me that each bit of plaintext would only have to diffuse to more than twice the number of bits in the small block size for the large block cipher to be as secure."
- W T Shaw "It is not necessary that all parts of the block be affected by changing a single bit in the data."
- Stefan Lucks "It is pretty easy to build block ciphers with huge blocks from conventional cryptographic primitives . . . ."
- W T Shaw "Like anything else new and/or different, large block ciphers need study to see if there are basic flaws."
- Wayne Schlitt "You want to be able to _analyze_ a cipher in order to _prove_ that it is strong."
- W T Shaw "That is the idea, to make it hard to correlate the data. A strong cipher should have this characteristic."
- Pat Morin "Look at the BEAR and LION ciphers developed by Ross Anderson and Eli Biham. They're fast, easy to implement, and have some nice security properties."


```
========
Path:
news.io.com!arlut.utexas.edu!cs.utexas.edu!howland.erols.net!surfnet.nl!swsbe6.switch.ch!scsing.switch.ch!elna.ethz.ch!caronni
From: caronni@tik.ethz.ch (Germano Caronni)
Newsgroups: sci.crypt
Subject: Ciphers with *Huge* Block Size ?
Date: 22 Aug 1996 17:18:35 GMT
Organization: Swiss Federal Institute of Technology (ETH), Zurich, CH
Lines: 33
Message-ID: <4vi4pb$fsq@elna.ethz.ch>
Reply-To: gec@acm.org
NNTP-Posting-Host: kom30-e.ethz.ch

Hello everybody,

current data encryption techniques usually encrypt a data stream or
small blocks of data. While this makes perfect sense for communication
where you need to encrypt and decrypt 'on the fly', it is perhaps not
the optimal solution for archiving, and generally not the strongest way
to secure data.  These are just idle thoughts, and I would be quite
interested to hear what you think about it.

Imagine a data file (or semantic entity, or whatever) being encrypted
as a whole, which means that each of the output bits depends on each of
the input bits (and naturally the secret key). This way, many kinds of
attacks may be a bit less feasible e.g. imagine doing a known plaintext
attack on something which simply can not be known in its entity, but of
which certain parts are well known (e.g. contents of encapsulated IP
headers, or headers of archived and encrypted mail) -- or imagine doing
a linear CA attack when the relevant block size is about a Megabyte.

Current block cipher chaining methods (CBC, OFB, CFB, ...) do not offer
this property, the best result is that the last encrypted 'block'
depends on all previous elements.
```

Comments to this one?

Germano

--
<...cookie space for rent...>

Germano Caronni caronni@tik.ee.ethz.ch http://www.tik.ee.ethz.ch/~caronni
PGP-Key-ID:7B7AE5E1 gec@acm.org 997C6DC4AF930A5D2D5D6AEAA196C33B

========
Path:
news.io.com!arlut.utexas.edu!cs.utexas.edu!bcm.tmc.edu!pendragon!news.msfc.nasa.gov!newsfeed.internetmci.com!howland.erols.net!math.ohio-state.edu!jussieu.fr!oleane!plug.news.pipex.net!pipex!hole.news.pipex.net!pipex!news.ukpats.org.uk!lade.news.pipex.net!pipex!tube.news.pipex.net!pipex!usenet
From: george.barwood@dial.pipex.com (George Barwood)
Newsgroups: sci.crypt
Subject: Re: Ciphers with *Huge* Block Size ?
Date: Thu, 22 Aug 1996 18:22:16 GMT
Organization: UUNet PIPEX server (post doesn't reflect views of UUNet PIPEX)
Lines: 22
Message-ID: <4vi8eg$b36@tube.news.pipex.net>
References: <4vi4pb$fsq@elna.ethz.ch>
NNTP-Posting-Host: al077.du.pipex.com
X-Newsreader: Forte Free Agent 1.0.82

caronni@tik.ethz.ch (Germano Caronni) wrote:

>Imagine a data file (or semantic entity, or whatever) being encrypted
>as a whole, which means that each of the output bits depends on each of
>the input bits (and naturally the secret key). This way, many kinds of
>attacks may be a bit less feasible e.g. imagine doing a known plaintext
>attack on something which simply can not be known in its entity, but of
>which certain parts are well known (e.g. contents of encapsulated IP
>headers, or headers of archived and encrypted mail) -- or imagine doing
>a linear CA attack when the relevant block size is about a Megabyte.

I think that the computation cost would be proportional to at least
the square of the block size, which for a 1Meg block might be
excessive. Most block ciphers try to be secure while running as fast
as possible which explains typical block sizes of 64-128 bits.
(Disclaimer: I am not an expert - the above may be rubbish)

========
Path:
news.io.com!arlut.utexas.edu!cs.utexas.edu!chi-news.cic.net!newspump.sol.net!nntp04.primenet.com!nntp.primenet.com!mr.net!news.sgi.com!enews.sgi.com!news.mathworks.com!newsfeed.internetmci.com!in2.uu.net!news.abs.net!news.synapse.net!tanda!marc
From: marc@tanda.on.ca (Marc Thibault)
Newsgroups: sci.crypt
Subject: Re: Ciphers with *Huge* Block Size ?
Message-ID: <5J54sD1w165w@tanda.on.ca>
Date: Fri, 23 Aug 96 08:00:51 EDT
References: <4vi8eg$b36@tube.news.pipex.net>
Distribution: world
Organization: Tanda
Lines: 35

george.barwood@dial.pipex.com (George Barwood) writes:
> I think that the computation cost would be proportional to at least
> the square of the block size, which for a 1Meg block might be

That's not intuitively obvious. As the block size goes up it should
be possible to reduce the complexity of the algorithm and still
maintain strength. If this is the case, the computational cost per
byte would go down. On the other hand, the cost of cryptanalysis
could go up dramatically with block size.

It is odd, though, that full-file algorithms aren't among the pack
and that there doesn't seem to be any literature on them - even to
explain why they are a bad idea.

I once put together a little code that treated a full file to a
reversible shuffling algorithm followed by adding a strong PRN
(really big random integer the same size as the file) and then
another shuffling. The shuffling and PRN were functions of the
variable-length key. It was in APL/360 and long gone, so I can't
offer any source.

One of the interesting things about this approach was that, since the
shuffling is effectively a variable-length transposition, a
single-bit change in the length of the file had dramatic effects on
the output. I think I could have made it much more interesting by
salting the file with something random in both content and length,
but I was young and ignorant at the time.
Cheers,
Marc

This is not a secure channel - Assume Nothing

http://www.hookup.net/~marct
Key fingerprint = 76 21 A3 B2 41 77 BC E8 C9 1C 74 02 80 48 A0 1A

========
Path:
news.io.com!insync!uuneo.neosoft.com!news.uh.edu!swrinde!howland.erols.net!nntp04.primenet.com!nntp.primenet.com!news.cais.net!news.abs.net!aplcenmp!netnews.jhuapl.edu!usenet
From: Bryan Olson
Newsgroups: sci.crypt
Subject: Re: Ciphers with *Huge* Block Size ?
Date: Thu, 22 Aug 1996 15:30:14 -0400
Organization: Johns Hopkins Applied Physics Lab
Lines: 38
Message-ID: <321CB546.7AF8@jhuapl.edu>
References: <4vi4pb$fsq@elna.ethz.ch>
Reply-To: Bryan_Olson@jhuapl.edu
NNTP-Posting-Host: snake.jhuapl.edu
Mime-Version: 1.0
Content-Type: text/plain; charset=us-ascii
Content-Transfer-Encoding: 7bit
X-Mailer: Mozilla 3.0 (WinNT; I)

Germano Caronni wrote:

> current data encryption techniques usually encrypt a data stream or
> small blocks [...] perhaps not the optimal solution for archiving,
> and generally not the strongest way to secure data.

> Imagine a data file [...] encrypted as a whole [...]
> each of the output bits depends on each of
> the input bits (and naturally the secret key). This way, many kinds of
> attacks may be a bit less feasible e.g. imagine doing a known plaintext
> attack on something which simply can not be known in its entity, but of
> which certain parts are well known (e.g. contents of encapsulated IP
> headers, or headers of archived and encrypted mail) -- or imagine doing
> a linear CA attack when the relevant block size is about a Megabyte.
>
> Current block cipher chaining methods (CBC, OFB, CFB, ...) do not offer
> this property, the best result is that the last encrypted 'block'
> depends on all previous elements.
>

While block size should be large enough to foil exhaustive search
over blocks, there's little evidence that it needs to be larger.

If you just want every bit to depend on every other, there are
chaining techniques which do so using conventional block ciphers.
For example:

Compute a hash of every block except the last. Encrypt the last
block using the hash as an initialization vector. Use the
ciphertext of the last block as an initialization vector for
the rest of the message, using CFB or CBC mode.

Now every bit depends on every other, and the whole thing can
be reversed without storing an initialization vector. I think
Peter Gutman used a technique similar to this in his SFS
(Secure File System).

--Bryan

========
Path:
news.io.com!arlut.utexas.edu!cs.utexas.edu!swrinde!howland.erols.net!cam-news-hub1.bbnplanet.com!news.mathworks.com!newsfeed.internetmci.com!in3.uu.net!news.injersey.com!news
From: John Michener
Newsgroups: sci.crypt
Subject: Re: Ciphers with *Huge* Block Size ?
Date: 22 Aug 1996 22:28:00 GMT
Organization: Asbury Park Press, Inc.
Lines: 20
Message-ID: <4vimtg$ak3@news.injersey.com>
References: <4vi4pb$fsq@elna.ethz.ch>
NNTP-Posting-Host: ppp064-trnt.injersey.com
Mime-Version: 1.0
Content-Type: text/plain; charset=us-ascii
Content-Transfer-Encoding: 7bit
X-Mailer: Mozilla 1.22KIT (Windows; U; 16bit)

Look at the construction of SP block codes. They typically
involve the successive application of a diffusion mechanism and a
set of S boxes. View the primitive block code, DES, as a very
large S-Box. Encrypt the the data with DES to mix all the bits
within consecutive 8 byte blocks. Transpose the data either
bitwise, one bit per subsequent 64 bit block to expand the
diffusion 64X, or 1 byte ber 8 byte block to expand the diffusion
8 X. Repeat the process. When the expansion fills the
super-block, repeat the process at least as many times as you have
already done. (hopefully with different DES / block keys). I am
afraid that the process will be very slow, and it is not obvious
that the cryptographic efficiency is particularily high. 3DES is
uncrackable anyway, and the super large block size certainly
creates problems in error diffusion and propagation.

To do the job right, you have to analyze what is happening from
the respect of known cryptanalytic techniques, such as linear or
differential cryptanalysis. The approach I mentioned above is
very rough.

```
========
Path:
news.io.com!news2.cais.net!news.cais.net!hunter.premier.net!news-res.gsl.net!news.gsl.net!news.mathworks.com!newsfeed.internetmci.com!in3.uu.net!ott.istar!istar.net!van.istar!west.istar!strat.enernet.com!cuugnet!not-for-mail
From: millerl@cuug.ab.ca (Lloyd Miller)
Newsgroups: sci.crypt
Subject: Re: Ciphers with *Huge* Block Size ?
Date: 22 Aug 1996 17:16:48 -0600
Organization: Calgary UNIX Users' Group
Lines: 16
Message-ID: <4vipp1$2j8@hp.cuug.ab.ca>
References: <4vi4pb$fsq@elna.ethz.ch>
NNTP-Posting-Host: hp.cuug.ab.ca
X-Newsreader: TIN [version 1.2 PL2]

Germano Caronni (caronni@tik.ethz.ch) wrote:
: Imagine a data file (or semantic entity, or whatever) being encrypted
: as a whole, which means that each of the output bits depends on each of
: the input bits (and naturally the secret key). This way, many kinds of
: attacks may be a bit less feasible e.g. imagine doing a known plaintext
: attack on something which simply can not be known in its entity, but of
: which certain parts are well known (e.g. contents of encapsulated IP
: headers, or headers of archived and encrypted mail) -- or imagine doing
: a linear CA attack when the relevant block size is about a Megabyte.

: Current block cipher chaining methods (CBC, OFB, CFB, ...) do not offer
: this property, the best result is that the last encrypted 'block'
: depends on all previous elements.

You just do two passes with a OFB type system. Do the second pass
backwards. I think one of the DOS device encryptors does this.


========
Path:
news.io.com!arlut.utexas.edu!cs.utexas.edu!howland.erols.net!newsxfer2.itd.umich.edu!bloom-beacon.mit.edu!pad-thai.cam.ov.com!gza-client1.cam.ov.com!not-for-mail
From: don@cam.ov.com (Donald T. Davis)
Newsgroups: sci.crypt
Subject: Re: Ciphers with *Huge* Block Size ?
Date: 22 Aug 1996 20:49:48 -0400
Organization: OpenVision Technologies, Inc.
Lines: 16
Message-ID: <4viv7c$6rk@gza-client1.cam.ov.com>
References: <4vi4pb$fsq@elna.ethz.ch>
NNTP-Posting-Host: gza-client1.cam.ov.com

gec@acm.org writes:
> Imagine a data file (or semantic entity, or whatever) being encrypted
> as a whole, which means that each of the output bits depends on each of the
> input bits (and naturally the secret key). ... Current block cipher chaining
> methods (CBC, OFB, CFB, ...) do not offer this property, the best result is
> that the last encrypted 'block' depends on all previous elements.

before encrypting for the archive, prepend the file's
message-digest to the plaintext. this will make every
ciphertext bit depend on every plaintext bit. it won't
work on connections or streamed data, but should work
fine for archives and file encryption.

                                -don davis, boston


========
Path:
news.io.com!arlut.utexas.edu!cs.utexas.edu!math.ohio-state.edu!uwm.edu!newsspool.doit.wisc.edu!news.doit.wisc.edu!news
From: Medical Electronics Lab
Newsgroups: sci.crypt
Subject: Re: Ciphers with *Huge* Block Size ?
Date: Fri, 23 Aug 1996 11:20:55 -0500
Organization: Dept. Neurophysiology, U. Wisconsin
Lines: 28
Message-ID: <321DDA67.5952@neurophys.wisc.edu>
References: <4vi4pb$fsq@elna.ethz.ch>
NNTP-Posting-Host: pcaf.neurophys.wisc.edu
Mime-Version: 1.0
Content-Type: text/plain; charset=us-ascii
Content-Transfer-Encoding: 7bit
X-Mailer: Mozilla 2.02 (WinNT; I)

Germano Caronni wrote:
> Imagine a data file (or semantic entity, or whatever) being encrypted
> as a whole, which means that each of the output bits depends on each of
> the input bits (and naturally the secret key). This way, many kinds of
> attacks may be a bit less feasible e.g. imagine doing a known plaintext
> attack on something which simply can not be known in its entity, but of
> which certain parts are well known (e.g. contents of encapsulated IP
> headers, or headers of archived and encrypted mail) -- or imagine doing
> a linear CA attack when the relevant block size is about a Megabyte.
>
> Current block cipher chaining methods (CBC, OFB, CFB, ...) do not offer
> this property, the best result is that the last encrypted 'block'
> depends on all previous elements.
```

>
> Comments to this one?

One comment pointed out that the encryption time expands with the
block size. I did look at this once (very briefly) with the idea
of using blocks on the order of 32K or so using elliptic curve math.
It can be done. It would take a depressingly long time to encrypt
a single block, many minutes on a 400 MHz Alpha (as in >100). It
might be possible to use many processors in parallel and go faster,
but the expense is ridiculous and the enhancement of security
miniscule compared to other options. It was fun to think about
tho ;-)

Patience, persistence, truth,
Dr. mike

========
Path:
news.io.com!insync!news.ios.com!news2.cais.net!news.cais.net!tezcat!cam-news-hub1.bbnplanet.com!news.mathworks.com!newsfeed.internetmci.com!in3.uu.net!info.htcomp.net!NewsWatcher!user
From: wtshaw@htcomp.net (W T Shaw)
Newsgroups: sci.crypt
Subject: Re: Ciphers with *Huge* Block Size ?
Date: 23 Aug 1996 18:44:23 GMT
Organization: Another Netscape News Server User
Lines: 23
Message-ID: <wtshaw-2308961347300001@207.17.188.113>
References: <4vi4pb$fsq@elna.ethz.ch> <321DDA67.5952@neurophys.wisc.edu>
NNTP-Posting-Host: 207.17.188.113

In article <321DDA67.5952@neurophys.wisc.edu>, Medical Electronics Lab
<rosing@neurophys.wisc.edu> wrote:
>
> One comment pointed out that the encryption time expands with the
> block size. I did look at this once (very briefly) with the idea
> of using blocks on the order of 32K or so using elliptic curve math.
> It can be done. It would take a depressingly long time to encrypt
> a single block, many minutes on a 400 MHz Alpha (as in >100). It
> might be possible to use many processors in parallel and go faster,
> but the expense is ridiculous and the enhancement of security
> miniscule compared to other options. It was fun to think about
> tho ;-)
>
This depends on algorithm; the one I use with large blocks is pleasingly
fast, not the speed of light, but a happy medium that is still slow enough
to let you watch the dynamic processing being done: it's a great demo that
I enjoy doing.
/\
wtshaw@htcomp.net   Mac Crypto Programs
 You should at least know how to use ROT13.
"Fhpprff vf n Wbhearl, Abg n Qrfgvangvba."
          http://www.htcomp.net/wts/
\/

========
Path: news.io.com!usenet
From: Terry Ritter
Newsgroups: sci.crypt
Subject: Re: Ciphers with *Huge* Block Size ?
Date: Fri, 23 Aug 1996 11:51:31 -0500
Organization: Ritter Software Engineering
Lines: 72
Message-ID: <321DE177.2193@io.com>
Reply-To: ritter@io.com
NNTP-Posting-Host: dialup-01-111.austin.io.com
Mime-Version: 1.0
Content-Type: text/plain; charset=us-ascii
Content-Transfer-Encoding: 7bit
X-Mailer: Mozilla 3.0 (Win95; I)

In <321CB546.7AF8@jhuapl.edu> Bryan Olson <Bryan_Olson@jhuapl.edu>
 writes:

>Germano Caronni wrote:
>
>> current data encryption techniques usually encrypt a data stream or
>> small blocks [...] perhaps not the optimal solution for archiving,
>> and generally not the strongest way to secure data.
>
>> Imagine a data file [...] encrypted as a whole [...]
>> each of the output bits depends on each of
>> the input bits (and naturally the secret key). This way, many kinds of
>> attacks may be a bit less feasible e.g. imagine doing a known plaintext
>> attack on something which simply can not be known in its entity, but of
>> which certain parts are well known (e.g. contents of encapsulated IP
>> headers, or headers of archived and encrypted mail) -- or imagine doing
>> a linear CA attack when the relevant block size is about a Megabyte.

 I have been working on this for some time. I have developed
 Fenced DES and Variable Size Block Cipher technologies, both of
 which are patent pending, both of which are described in
 excruciating detail on my pages:

      http://www.io.com/~ritter/

>While block size should be large enough to foil exhaustive search
>over blocks, there's little evidence that it needs to be larger.

First, in cryptography, it is *not* appropriate to demand proof of weakness before adding strength to a cipher. This is a fundamental difference between real cryptography and ivory-tower Science.

Next, it is clear that codebook attacks can start to be effective when we get two ciphertext blocks which are the same. If we assume an absolutely flat plaintext distribution (we assume cipher block chain (CBC) mode, or other plaintext randomizer) and 64-bit blocks, we can expect to see this with about 2**32 blocks, and rapidly worsening thereafter. This is 2**35 bytes or about 34 GB. While this may seem huge now, we will soon see disk drives of this size, and in two decades (a reasonable time to expect a new cipher design to last), 34 GB will be humorously small. Yes, we can re-key and continue for larger data, but if we have to consider this (and any modern design must), we *do* have evidence that the block size could usefully be larger.

In fact, the whole need to use CBC (or other plaintext randomization) is based on the small amount of language entropy in 8 character blocks: If we assume 1.5 bits of entropy per character, we are looking at 12 bits of entropy -- effectively only 4096 choices (times 8, if we consider each position in the block). No wonder that electronic codebook (ECB) mode is deprecated.

On the other hand, a 200-byte block of plaintext language should have about 300 bits of entropy, so we could expect to find two the same in about 2**150 blocks. Thus, if we have a large block, ECB becomes practical, which could be useful in certain situations, and this *is* evidence that the ciphering block size could usefully be larger.

And we have yet to consider the potential security advantage of using dynamically pseudo-random size blocks -- along with random padding -- in messages. (I have implemented practical ciphers with dynamically-variable blocks, described them on sci.crypt, and archived the messages on my pages.) The lack of fixed block boundaries presents serious problems for any current attack on a block cipher. So, yet again, we see evidence that ciphering block size could usefully be larger, and also could usefully be dynamically variable.

Terry Ritter ritter@io.com http://www.io.com/~ritter/

========
Path:
news.io.com!arlut.utexas.edu!cs.utexas.edu!math.ohio-state.edu!uwm.edu!newsfeed.internetmci.com!newsxfer2.itd.umich.edu!uunet!in2.uu.net!news.new-york.net!news.columbia.edu!news.cs.columbia.edu!versed.smarts.com!usenet
From: Jerry Leichter
Newsgroups: sci.crypt
Subject: Re: Ciphers with *Huge* Block Size ?
Date: Fri, 23 Aug 1996 14:05:51 -0400
Organization: System Management ARTS
Lines: 89
Message-ID: <321DF2FF.7114@smarts.com>
References: <321DE177.2193@io.com>
NNTP-Posting-Host: just.smarts.com
Mime-Version: 1.0
Content-Type: text/plain; charset=us-ascii
Content-Transfer-Encoding: 7bit
X-Mailer: Mozilla 3.0b5aGold (X11; I; SunOS 5.5 sun4d)
To: ritter@io.com

Terry Ritter wrote:
> [Interesting examples of uses for ciphers with large, or even
> variable, block sizes.]

The counter-arguments that have held sway are more application-oriented and historical than theoretical:

1. Hardware implementations of large block sizes are more difficult. *How* difficult, of course, changes as technology advances. Certainly when DES was first defined, it was at the limits of reasonable-cost hardware. Today you could certainly do a much "wider" chip.

2. The biggest demand is for encryption of communication streams. (We know how to do physical security, so keeping disks safe is *relatively* easier. But it's very difficult to keep "wires" secure except in unusual circumstances.)

   This adds a number of constraints. For example, stream mode operation is highly desireable; you can't afford to buffer large amounts of information just for encryption. (The original question in this thread concerned encryption modes in which the output depended on the *entire* input, no matter how large. Such a mode would be entirely unusable in a communications

framework.)

3. The larger the block, the larger the average space that is wasted to fill out short blocks. In a communications framework, many messages are quite short. A really large block could end up devoting more space to fill than to useful message!

4. All other things being equal, a large-block cipher is probably going to be slower than a small-block cipher on a per-bit basis: If each bit of the output depends on each bit of the input, then the minimum possible operations per bit doubles when you double the block size. (This kind of argument is necessarily vague and incomplete, since with a larger block size you may be able to get away with, say, fewer rounds for equivalent security. But I know of no ciphers that actually come out ahead for large block sizes on this basis.)

5. Many of these issues grow less important with each passing year as chips get faster (though encryption of traffic on the highest speed lines remains problematical - transmission data rates are going up faster than CPU speeds these days, and the trend lines indicate that this will continue. If it does, we'll have to go for parallelism of some sort - perhaps just pipelining is enough - and that's much easier to do with smaller block sizes.) However, there is an existing infra-structure now. Block sizes are often "frozen in" to existing designs and difficult and expensive to change. That's one reason people like triple-DES: It can be used as a drop-in replacement for DES, with effects only on the relatively low-frequency activity of key distribution.

6. Ritter's arguments are strongest when applied to file (or disk) encryption, and larger block sizes might be very appropriate there (up to some limit, since you don't want the cipher block size to force you to transfer significantly more data to and from the disk). However, there is also a strong pressure to standardize on one encryption technology - DES today, who-knows-what tomorrow. If one decides that it's a good idea to use the same cryptosystem for both files and communications (which has its pluses, like the ability to use fast encryption chips developed for communications in disk controllers), then constraints on the communications side "leak through" to the file side, even if they aren't really relevant there.

   (Of course, from a security point of view, the more strong cryptosystems you have, the better. All the available evidence indicates that NSA produces many different cryptosystems for different purposes. That's just prudent design. The same is actually true for any component with security implications - a security hole in, say, Netscape implies a vulnerability on the part of almost all Web users these days. Nevertheless, the pressures for standardization are so great that they are often impossible to resist.)

-- Jerry

[Ritter responds.]

========
Path:
news.io.com!insync!uuneo.neosoft.com!imci3!newsfeed.internetmci.com!in2.uu.net!info.htcomp.net!NewsWatcher!user
From: wtshaw@htcomp.net (W T Shaw)
Newsgroups: sci.crypt
Subject: Re: Ciphers with *Huge* Block Size ?
Date: 23 Aug 1996 19:00:57 GMT
Organization: Another Netscape News Server User
Lines: 73
Message-ID: <wtshaw-2308961403560001@207.17.188.113>
References: <321DE177.2193@io.com>
NNTP-Posting-Host: 207.17.188.113

In article <321DE177.2193@io.com>, ritter@io.com wrote:
>
> I have been working on this for some time. I have developed
> Fenced DES and Variable Size Block Cipher technologies, both of
> which are patent pending, both of which are described in
> excruciating detail on my pages:
>
> http://www.io.com/~ritter/
>
>
> >While block size should be large enough to foil exhaustive search
> >over blocks, there's little evidence that it needs to be larger.
>
> First, in cryptography, it is *not* appropriate to demand proof of
> weakness before adding strength to a cipher. This is a fundamental
> difference between real cryptography and ivory-tower Science.
>

> Next, it is clear that codebook attacks can start to be effective
> when we get two ciphertext blocks which are the same. If we assume
> an absolutely flat plaintext distribution (we assume cipher block
> chain (CBC) mode, or other plaintext randomizer) and 64-bit blocks,
> we can expect to see this with about 2**32 blocks, and rapidly
> worsening thereafter. This is 2**35 bytes or about 34 GB. While
> this may seem huge now, we will soon see disk drives of this size,
> and in two decades (a reasonable time to expect a new cipher design
> to last), 34 GB will be humorously small. Yes, we can re-key and
> continue for larger data, but if we have to consider this (and any
> modern design must), we *do* have evidence that the block size
> could usefully be larger.
>
> In fact, the whole need to use CBC (or other plaintext randomization)
> is based on the small amount of language entropy in 8 character
> blocks: If we assume 1.5 bits of entropy per character, we are
> looking at 12 bits of entropy -- effectively only 4096 choices
> (times 8, if we consider each position in the block). No wonder
> that electronic codebook (ECB) mode is deprecated.
>
> On the other hand, a 200-byte block of plaintext language should
> have about 300 bits of entropy, so we could expect to find two the
> same in about 2**150 blocks. Thus, if we have a large block, ECB
> becomes practical, which could be useful in certain situations,
> and this *is* evidence that the ciphering block size could
> usefully be larger.
>
> And we have yet to consider the potential security advantage of
> using dynamically pseudo-random size blocks -- along with random
> padding -- in messages. (I have implemented practical ciphers
> with dynamically-variable blocks, described them on sci.crypt,
> and archived the messages on my pages.) The lack of fixed block
> boundaries presents serious problems for any current attack on
> a block cipher. So, yet again, we see evidence that ciphering
> block size could usefully be larger, and also could usefully be
> dynamically variable.
>
> ---
> Terry Ritter ritter@io.com http://www.io.com/~ritter/

By keeping block size as small as is common, the strengths of the
algorithms are crippled so as to make it difficult to make them
increasingly stronger.

Close examination of Terry's algorithms will reveal the achievement
therein. Good, new algorithms, such as these, would demand entirely new
cryptoanalytical efforts, thus making them less vulnerable to immediate
attack. The quantum leap in size of the blocks also greatly complicates
the matter for analysis. There is tremendous potential for commercial
application of these algorithms.
/\
wtshaw@htcomp.net Mac Crypto Programs
You should at least know how to use ROT13.
"Fhpprff vf n Wbhearl, Abg n Qrfgvangvba."
http://www.htcomp.net/wts/
\/

========
Path:
news.io.com!arlut.utexas.edu!cs.utexas.edu!howland.erols.net!newsfeed.internetmci.com!in3.uu.net!news.abs.net!aplcenmp!netnews.jhuapl.edu!usenet
From: Bryan Olson
Newsgroups: sci.crypt
Subject: Re: Ciphers with *Huge* Block Size ?
Date: Mon, 26 Aug 1996 17:25:09 -0400
Organization: Johns Hopkins Applied Physics Lab
Lines: 27
Message-ID: <32221635.1686@jhuapl.edu>
References: <321DE177.2193@io.com> <wtshaw-2308961403560001@207.17.188.113>
Reply-To: Bryan_Olson@jhuapl.edu
NNTP-Posting-Host: snake.jhuapl.edu
Mime-Version: 1.0
Content-Type: text/plain; charset=us-ascii
Content-Transfer-Encoding: 7bit
X-Mailer: Mozilla 3.0 (WinNT; I)

W T Shaw wrote:
>
> In article <321DE177.2193@io.com>, ritter@io.com (Terry Ritter)
> wrote:
> >
> > I have been working on this for some time. I have developed
> > Fenced DES and Variable Size Block Cipher technologies [...]

> Close examination of Terry's algorithms will reveal the achievement
> therein. Good, new algorithms, such as these, would demand entirely new
> cryptoanalytical efforts, thus making them less vulnerable to immediate
> attack. The quantum leap in size of the blocks also greatly complicates
> the matter for analysis. There is tremendous potential for commercial
> application of these algorithms.

I'd invite anyone who has closely examined Terry's "Fenced DES"
to comment on the attack I proposed in another thread of this
same topic. If I've understood Terry's algorithm correctly, the
attack should work, and I may write up a better exposition for

the research group.

The attack doesn't use any really new or novel methods, and the
large block seemsto make it easier. I still agree that large
blocks could have great security advantages, but I would not
trade per-bit confusion/diffusion for block size.

--Bryan

========
Path:
news.io.com!news2.cais.net!news.cais.net!tezcat!cam-news-hub1.bbnplanet.com!news.mathworks.com!news.kei.com!newsfeed.internetmci.com!in2.uu.net!news.abs.net!aplcenmp!netnews.jhuapl.edu!usenet
From: Bryan Olson
Newsgroups: sci.crypt
Subject: Re: Ciphers with *Huge* Block Size ?
Date: Fri, 23 Aug 1996 15:46:41 -0400
Organization: Johns Hopkins Applied Physics Lab
Lines: 40
Message-ID: <321E0AA1.4517@jhuapl.edu>
References: <321DE177.2193@io.com>
Reply-To: Bryan_Olson@jhuapl.edu
NNTP-Posting-Host: snake.jhuapl.edu
Mime-Version: 1.0
Content-Type: text/plain; charset=us-ascii
Content-Transfer-Encoding: 7bit
X-Mailer: Mozilla 3.0 (WinNT; I)

Terry Ritter wrote:
>
> In <321CB546.7AF8@jhuapl.edu> Bryan Olson <Bryan_Olson@jhuapl.edu>
> writes:
>
>
> >While block size should be large enough to foil exhaustive search
> >over blocks, there's little evidence that it needs to be larger.
>
> First, in cryptography, it is *not* appropriate to demand proof of
> weakness before adding strength to a cipher. This is a fundamental
> difference between real cryptography and ivory-tower Science.
>

No argument there (well, except for the academic-bashing).
I'm not against large blocks. I'm pointing out that no one
has shown they're more secure. The advantages that Germano
cited, and most that you cite, can be achieved by chaining
smaller blocks.

I know you've devised several large-block ciphers, but your
security results are far too speculative. One of the basic
principles in block-cipher design is that each key bit and
data bit should interact with the others many times in
internal transformations. Large-block ciphers seem to
require more operations to achieve the same level of
diffusion, and consequently most settle for less.

I suspect that there is a security advantage to larger blocks.
The same level of diffusion ought to be harder to reverse in
a larger block. No, I can't prove it.

Finally, I don't like designs which work extensively on
isolated sections of the key and/or data space. DES shows
good design in that all the key bits come into play
repeatedly. "Triple DES" and "Fenced DES" choose to trust
in some unknown properties of the cipher, rather than follow
the established principles of its design.

--Bryan

========
Path: news.io.com!usenet
From: ritter@io.com (Terry Ritter)
Newsgroups: sci.crypt
Subject: Re: Ciphers with *Huge* Block Size ?
Date: Sat, 24 Aug 1996 07:55:20 GMT
Lines: 140
Message-ID: <4vmcm7$8jd@nntp-1.io.com>
References: <321DE177.2193@io.com> <321E0AA1.4517@jhuapl.edu>
NNTP-Posting-Host: dialup-01-088.austin.io.com
X-Newsreader: Forte Free Agent 1.0.82

 In <321E0AA1.4517@jhuapl.edu> Bryan Olson <Bryan_Olson@jhuapl.edu>
 writes:

>I'm not against large blocks. I'm pointing out that no one
>has shown they're more secure. The advantages that Germano
>cited, and most that you cite, can be achieved by chaining
>smaller blocks.

 The very need to chain small blocks is itself mainly a consequence
 of tiny block size, and its general use an admission of the known
 weakness of small blocks. With a large block, we generally have
 sufficient language entropy to safely cipher in electronic

codebook mode (ECB). (One might argue for chaining as a modification-detection mechanism, but nowadays a cryptographic hash handles this.)

>I know you've devised several large-block ciphers, but your
>security results are far too speculative.

Fine. What would you like to see?

Are you convinced about the strength of DES? If so, what evidence convinced you? If you are instead convinced by the testimony of various crypto gods, there is not much I care to do about that.

>One of the basic
>principles in block-cipher design is that each key bit and
>data bit should interact with the others many times in
>internal transformations.

No, this is not strictly true.

The principle you are looking for is called "avalanche," and states a *goal* that every output bit should be "affected" by every input bit. (This is no more than a consequence of the observation that a block cipher is a machine which attempts to emulate a Simple Substitution table of impractical size. Every Simple Substitution will avalanche.)

The testable result is that a change on any input bit should change (almost) half of the output bits, on average. Any implication that one can only get such a result if bits *explicitly* interact "many times" is not warranted, as far as I know.

>Large-block ciphers seem to
>require more operations to achieve the same level of
>diffusion, and consequently most settle for less.

No. Modern mixing guarantees distribution of any input change to every substitution in the next layer in n log n operations. (This is either log n layers of n-byte multi-precision mixing, or log n layers of byte-by-byte mixing.) Feistel mixing gives no such guarantees, and, indeed, virtually requires experiment to decide how many rounds are "enough." (Authors of similar Feistel ciphers often let the user decide how many rounds are enough, as though users will have the background to do this.)

The use of keyed invertible substitutions guarantees that *any* change in a substitution input value *will* avalanche the output from the substitution. (The use of keyed substitutions also protects against Differential and Linear attacks, since the transformation is keyed and not known externally.) Again, Feistel operations have no guarantees.

The separate use of balanced, complete mixing layers and a "fencing" layer of keyed, invertible substitutions guarantees avalanche across large blocks. The use of another set of layers protects against various "meet in the middle" or "fix in the middle" attacks (and may not always be necessary). The use of a fencing layer before linear mixing protects the mixing, provided that the subsequent ciphering layers have not been exposed.

>Finally, I don't like designs which work extensively on
>isolated sections of the key and/or data space.

Then you should *love* my ciphers. I place heavy emphasis on keying substitution tables. This is done by shuffling under the control of a substantial RNG (typically, a 992 bit Additive RNG). The RNG is initialized by 32 31-bit CRC's using different primitives each time. This means that every entry in every table is affected by every bit in the key.

Then, the guaranteed mixing in these designs means that every input bit "affects" *every* substitution (each of which has been affected by every key bit) in two of three layers.

I can provide much *better* guarantees of key interaction than anything I've seen on the DES design.

>DES shows
>good design in that all the key bits come into play
>repeatedly.

This sounds about like: "DES shows good design because it is very like DES, which is known to be a good design." Not only is this circular, but we do *not* *know* that DES *is* a good design. In fact, we actually do know that DES is obsolete.

While DES does repeatedly use the key bits, the DES design does *not* show that this is the way ciphering *must* be done, or even *should* be done, to achieve strength. An example of one design is not a good reference for ciphers built on fundamentally different

principles of operation, even if similar results are desired.

As a matter of fact, this common design concept "let's just make something pretty much like DES, only a little better" is very seriously flawed due to a lack of known mathematical basis for Feistel-type designs. The fact that many do this does not make it a good design technique.

>"Triple DES" and "Fenced DES" choose to trust
^^^^^^^^^^
("Variable Size Block Ciphers"?)

>in some unknown properties of the cipher, rather than follow
>the established principles of its design.

Your so-called "established principles" do not, alas, begin to provide the mathematical guarantees of operation provided by Fencing and Mixing cipher designs. It seems strange you would bring this up, for it is the conventional wisdom of Feistel ciphering which is weak here, not these designs.

>
>--Bryan
>

Terry Ritter ritter@io.com http://www.io.com/~ritter/

========
Path:
news.io.com!arlut.utexas.edu!cs.utexas.edu!howland.erols.net!cam-news-hub1.bbnplanet.com!cpk-news-hub1.bbnplanet.com!cpk-news-feed1.bbnplanet.com!netnews.jhuapl.edu!usenet
From: Bryan Olson
Newsgroups: sci.crypt
Subject: Re: Ciphers with *Huge* Block Size ?
Date: Sun, 25 Aug 1996 21:36:45 -0400
Organization: Johns Hopkins Applied Physics Lab
Lines: 229
Message-ID: <3220FFAD.27BB@jhuapl.edu>
References: <321DE177.2193@io.com> <321E0AA1.4517@jhuapl.edu>
<4vmcm7$8jd@nntp-1.io.com>
Reply-To: Bryan_Olson@jhuapl.edu
NNTP-Posting-Host: snake.jhuapl.edu
Mime-Version: 1.0
Content-Type: text/plain; charset=us-ascii
Content-Transfer-Encoding: 7bit
X-Mailer: Mozilla 3.0 (WinNT; I)

Terry Ritter wrote:
>
> In <321E0AA1.4517@jhuapl.edu> Bryan Olson <Bryan_Olson@jhuapl.edu>
> writes:
>

> >I know you've devised several large-block ciphers, but your
> >security results are far too speculative.
>
> Fine. What would you like to see?
>

I'd like to see fewer proposed ciphers with much more careful analysis. I took another look at your "Fenced DES" on your web pages (I was happy to see you kept my critical posts too). You say,
"If we assume that the internal block mixing is indeed effective, the overall strength of 4x Fenced DES is at least four times that of 1x Fenced DES, for a total of 480 bits."

How carefully did you analyze your own cipher before you proposed it? Are you saying that exhausting a 480 bit space was the best attack you could come up with? Would that include an on-line selected plaintext attack?

(To follow the next section, readers will have to understand Ritter's "Fenced DES". See: http://www.io.com/~ritter/FENCED.HTM
)

You actually start on an attack that I think could be productive, but you seem to bring it up only to discount it.

(Of course, for substantial input changes, it is possible to generate values which will leave one or more of the DES operations unaffected, although this is quite unlikely to happen by chance. In the mixing transform, such a circumstance requires specific related 128-bit values on each of the two input ports, and these values are hidden behind the input substitutions. We could manage to keep a DES operation unaffected if we knew the arrangement of the values in all the input substitutions, but that uncertainty is part of the strength of the cipher. And it seems unlikely that we could tell from the outside that we were successful -- that fewer than four DES operations have avalanched -- because any avalanche will enter the output block mixing and so be

reflected over the whole width of the large block, with the
specific resulting values hidden by the output substitutions.)

In your block mixing functions, it seems that a change of one bit
position in either half of the input can change only the output bit
in the same position and the one to the immediate left (circularly).
Am I wrong on that? If so, the change between a plaintext A and A'
needed to make three of the internal DES applications receive the
same input are not all that "substantial". Even with the
substitutions we could certainly search for them.

I can pick four bytes that will be mixed only with each other in
both levels of pre-DES mixing. In all 2^32 four byte values, there
should be several (maybe 64, I haven't ground it out) pairs of A and A'
that result in a 0 difference for inputs of three of the DES
applications. I'll call A and A' a "match" if they have that property.

You say it "seems unlikely that we could tell from the outside that we
were successful" in finding such a match. If A and A' are a match, I
can immediately find many other matches by changing bytes which don't
interact with the critical four. That gives me a good space to search
for any statistical anomaly which would indicate a match.

So what anomaly do I search for?

Let's turn our attention to an output of the DES boxes. If I have a
match, three of them are the same under A as under A'. In your analysis
you assume the output of the fourth will be unrelated between A and A',
and this will make all the outputs appear unrelated. But what if just
some output bits from this S box turn out to be the under A and A', and
in a fortuitous combination. Specifically:

1. One of the eight bytes (on the byte boundary)
2. The two most significant bits in the byte to the right of 1.

If that happens, then all 4 DES outputs will have 10 bits positions
which are the same under A as under A'. The block mixing will preserve
8 of them, lined up with a single substitution at the end. Thus in the
output, we'll see four bytes, each three bytes apart, which are
identical
under A and A'.

By chance, the four identical bytes should appear once in 2^24 pairs A
and
A', for each of the 8 byte positions. Conditions 1 and two (above)
should
hold about once in 2^10 pairs. If they hold under a non-match, then the
four-byte coincidence still has the same probability, but if they hold
under a match, then the coincidence is a certainty.

So if my four critical bytes do _not_ force a match, I expect to see the
four-byte pattern one in 2^21 pairs. If they are a match, I'll see it
once in 2^7 pairs, where I form the pairs by holding these four input
bytes
constant and varying others (not immediately to their circular right).

So I have to search over a 2^32 space, and run a few hundred pairs for
each to find such a match. Once I've found it, I have a lot of
information about the four permutations which formed my four critical
bytes. Repeating the process to find all the matches over all 32
permutions, I should be able to break the initial permutations. Looking
at what happens in the output, I can probably infer a lot about the
final
permutations too.

That's as far as I'll take the attack for now. Perhaps I've
misunderstood something about the design; please let me know if any
of the steps I've described is impossible. I'll elaborate if parts of
my explanation are unclear. If as much as I've described can be done,
I certainly think it's a strong enough toe-hold for the analyst that
Fenced DES is dead.

Back to the thread...

> Are you convinced about the strength of DES?

Absolutely, positively not. I'm convinced of its weakness, and
I don't think it should be used as a component in a larger cipher.

> >One of the basic
> >principles in block-cipher design is that each key bit and
> >data bit should interact with the others many times in
> >internal transformations.
>
> No, this is not strictly true.
>
> The principle you are looking for is called "avalanche," and states
> a *goal* that every output bit should be "affected" by every input
> bit. (This is no more than a consequence of the observation that
> a block cipher is a machine which attempts to emulate a Simple
> Substitution table of impractical size. Every Simple Substitution
> will avalanche.)
>
> The testable result is that a change on any input bit should change
> (almost) half of the output bits, on average. Any implication that

> one can only get such a result if bits *explicitly* interact "many
> times" is not warranted, as far as I know.
>

No, the criteria I meant was the one I said. Avalanche is not enough
(and to be fair neither is my criteria; they're both necessary but not
sufficient). All the bits need to be effected in a complex way. If
sections of the key are used just once, it often allows the analyst to
partition the effect of those bits from the effects of others. That
is why meet-in-the-middle works, why your NxM DES failed, and how I
attacked Fenced DES.

>
> The use of keyed invertible substitutions guarantees that *any*
> change in a substitution input value *will* avalanche the output
> from the substitution. (The use of keyed substitutions also
> protects against Differential and Linear attacks, since the
> transformation is keyed and not known externally.) Again, Feistel
> operations have no guarantees.
>
> The separate use of balanced, complete mixing layers and a "fencing"
> layer of keyed, invertible substitutions guarantees avalanche
> across large blocks. The use of another set of layers protects
> against various "meet in the middle" or "fix in the middle"
> attacks (and may not always be necessary). The use of a fencing
> layer before linear mixing protects the mixing, provided that
> the subsequent ciphering layers have not been exposed.
>

I'll let you assess my attack before I argue the problems with that.

> >Finally, I don't like designs which work extensively on
> >isolated sections of the key and/or data space.
>
> Then you should *love* my ciphers.

Uh...

[...]
> I can provide much *better* guarantees of key interaction than
> anything I've seen on the DES design.
>

Obviously I disagree.

> >DES shows
> >good design in that all the key bits come into play
> >repeatedly.
>
> This sounds about like: "DES shows good design because it is
> very like DES, which is known to be a good design." Not only
> is this circular, but we do *not* *know* that DES *is* a good
> design. In fact, we actually do know that DES is obsolete.
>
> While DES does repeatedly use the key bits, the DES design does
> *not* show that this is the way ciphering *must* be done, or even
> *should* be done, to achieve strength. An example of one design
> is not a good reference for ciphers built on fundamentally different
> principles of operation, even if similar results are desired.
>

See also IDEA, Blowfish, and most other block ciphers which have
resisted attack.

> As a matter of fact, this common design concept "let's just make
> something pretty much like DES, only a little better" is very
> seriously flawed due to a lack of known mathematical basis for
> Feistel-type designs. The fact that many do this does not make it
> a good design technique.
>

Which side are you arguing? You're the one trying to use DES but add
some simple layers to make it stronger.

> Your so-called "established principles" do not, alas, begin to
> provide the mathematical guarantees of operation provided by
> Fencing and Mixing cipher designs. It seems strange you would
> bring this up, for it is the conventional wisdom of Feistel
> ciphering which is weak here, not these designs.
>
> Terry Ritter   ritter@io.com   http://www.io.com/~ritter/

I think I probably have a few numbers wrong in my analysis, but
not so far wrong as invalidate the attack on Fenced DES. If
the attack doesn't work, I'd like to know why. If it does,
well I have to respect you for showing your failures on your web
page.

--Bryan

========
Path: news.io.com!usenet
From: ritter@io.com (Terry Ritter)
Newsgroups: sci.crypt

Subject: Re: Ciphers with *Huge* Block Size ?
Date: Tue, 27 Aug 1996 18:05:14 GMT
Organization: Illuminati Online
Lines: 530
Message-ID: <4vvdbj$3de@anarchy.io.com>
References: <321DE177.2193@io.com> <321E0AA1.4517@jhuapl.edu>
<4vmcm7$8jd@nntp-1.io.com> <3220FFAD.27BB@jhuapl.edu>
NNTP-Posting-Host: dialup-01-146.austin.io.com
X-Newsreader: Forte Free Agent 1.0.82

In <3220FFAD.27BB@jhuapl.edu> Bryan Olson <Bryan_Olson@jhuapl.edu>
wrote:

>I'd like to see fewer proposed ciphers with much more careful
>analysis.

But no analysis is *possible* without a design. And new designs also are *not* *possible* without considering new basic structures. Some may fail, others may not. Big deal.

Frankly, I'd like to see the academic community examining a wide variety of different structures, categorizing them, and working out some common insights. But this *is* "dangerous work": Really new proposals can "fail" and damage a frail academic reputation.

Heaven help us all if there turns out to be a general attack on Feistel ciphering.

>I took another look at your "Fenced DES" on your web
>pages (I was happy to see you kept my critical posts too).

I believe in presenting both sides of a controversy. I also believe in documenting the mistakes of the past so as to hopefully reduce such mistakes in the future. It may well be that both the readers and the author will learn more from design failures than successes.

>You
>say,
> "If we assume that the internal block mixing is indeed
> effective, the overall strength of 4x Fenced DES is at least
> four times that of 1x Fenced DES, for a total of 480 bits."
>
>How carefully did you analyze your own cipher before you proposed
>it?

What kind of question is this? How careful is "careful"? How should I measure it (let me count the ways!).

I was about as careful as I would be in completing an exam, given various other obligations and resource limitations. I did the best that I could at the time, but certainly did not claim that the cipher was unbreakable. Considering that academia has failed to produce such a cipher, that would be nuts. The *whole* *point* in publication is for somebody else to find a flaw if it exists.

In producing software, I have found it useful to figuratively "reward" myself for finding bugs, despite the fact that each and every bug can be seen as my own programming "failure." The idea that humans can produce perfect programs is a goal, not reality. We instead must use strategies which minimize errors and then expose and fix those errors which do occur. But other than producing yet another Feistel cipher which is difficult or impossible to analyze, I am not aware of effective strategies to minimize errors in cipher design. Instead I accept the fact that bugs may exist in my ciphers, work at exposing those bugs and understanding the deeper ramifications of ones which are found.

If your implication is that any proposal which is flawed would not have been made if one was sufficiently "careful," I can list many "serious" academic proposals, from very respected people, which turned out -- in retrospect -- to be flawed. This is, in fact, the way we expect Science to progress. Rigid expectations of "care" are not appropriate to the world as I see it.

>Are you saying that exhausting a 480 bit space was the best
>attack you could come up with? Would that include an on-line
>selected plaintext attack?

Do you suggest that I am writing this stuff knowing that it has problems? If had come up with something I considered better, it would be there.

>(To follow the next section, readers will have to understand
>Ritter's "Fenced DES". See: http://www.io.com/~ritter/FENCED.HTM
>)
>
>You actually start on an attack that I think could be productive,
>but you seem to bring it up only to discount it.

I brought the attack up because it seemed obvious, and presented it as I did because it seemed unproductive with just a little more thought. Had I not brought it up, you might have accused me of hiding it. Obviously I can't win either way, which is OK, as long as we all see the same game.

> (Of course, for substantial input changes, it is possible to
> generate values which will leave one or more of the DES
> operations unaffected, although this is quite unlikely to
> happen by chance. In the mixing transform, such a circumstance
> requires specific related 128-bit values on each of the two
> input ports, and these values are hidden behind the input
> substitutions. We could manage to keep a DES operation
> unaffected if we knew the arrangement of the values in all the
> input substitutions, but that uncertainty is part of the
> strength of the cipher. And it seems unlikely that we could
> tell from the outside that we were successful -- that fewer
> than four DES operations have avalanched -- because any
> avalanche will enter the output block mixing and so be
> reflected over the whole width of the large block, with the
> specific resulting values hidden by the output substitutions.)
>
>In your block mixing functions, it seems that a change of one bit
>position in either half of the input can change only the output bit
>in the same position and the one to the immediate left (circularly).
>Am I wrong on that?

Just technically. We have a field; sometimes the field polynomial is activated and we get its bits too. Usually not.

>If so, the change between a plaintext A and A'
>needed to make three of the internal DES applications receive the
>same input are not all that "substantial". Even with the
>substitutions we could certainly search for them.

Hardly "certainly." Search implies recognition of difference. It would be easy to cycle through all combinations of four bytes and know that somewhere in there the combination(s) we wanted must exist. The problem is knowing when that occurred.

>I can pick four bytes that will be mixed only with each other in
>both levels of pre-DES mixing.

This doesn't take much "picking"; this is the situation in general. The description I would give is that each of the four bytes is distributed to each of the internal ciphers. Changing one bit of any one byte *is* guaranteed to change each of the inputs to each of internal ciphers. Changing bits in multiple bytes does not have this same guarantee.

>In all 2^32 four byte values, there
>should be several (maybe 64, I haven't ground it out) pairs of A and A'
>that result in a 0 difference for inputs of three of the DES
>applications. I'll call A and A' a "match" if they have that property.
>
>You say it "seems unlikely that we could tell from the outside that we
>were successful" in finding such a match. If A and A' are a match, I
>can immediately find many other matches by changing bytes which don't
>interact with the critical four. That gives me a good space to search
>for any statistical anomaly which would indicate a match.

If we *assume* that A and A' match, we can do lots of things. The issue is finding that match.

>So what anomaly do I search for?
>
>Let's turn our attention to an output of the DES boxes. If I have a
                                                        [box]
>match, three of them are the same under A as under A'.

Yes.

>In your analysis
>you assume the output of the fourth will be unrelated between A and A',
>and this will make all the outputs appear unrelated.

Yes.

>But what if just
>some output bits from this S box turn out to be the under A and A',
                                                        ^[same]
>and in a fortuitous combination. Specifically:
> 1. One of the eight bytes (on the byte boundary)
> 2. The two most significant bits in the byte to the right of 1.

You are proposing a situation where particular bit-position values occur "at random" from an internal DES operation. The probability of this would seem to be 1 in 2**10.

[Also note that DES is not an "S-box" but a "cipher." The difference is structural: S-boxes, as the term is used in Feistel ciphering (e.g., DES) are not constrained to be permutation functions -- that is, Simple Substitutions, but ciphers are.]

>If that happens, then all 4 DES outputs will have 10 bits positions
>which are the same under A as under A'.

Yes. If that happens.

>The block mixing will preserve
>8 of them, lined up with a single substitution at the end. Thus in the
>output, we'll see four bytes, each three bytes apart, which are
>identical
>under A and A'.

Generally, yes. Three levels of mixing poly will each tend to activate about half the time, though.

>By chance, the four identical bytes should appear once in 2^24
>pairs A and A', for each of the 8 byte positions.

Yes.

>Conditions 1 and two (above) should hold about once in 2^10 pairs.

Yes.

>If they hold under a non-match, then the four-byte coincidence
>still has the same probability, but if they hold
>under a match, then the coincidence is a certainty.

?

>So if my four critical bytes do _not_ force a match, I expect to
>see the
>four-byte pattern one in 2^21 pairs. If they are a match, I'll
>see it
>once in 2^7 pairs, where I form the pairs by holding these four
>input bytes
>constant and varying others (not immediately to their circular
>right).

!

>So I have to search over a 2^32 space, and run a few hundred pairs
>for each to find such a match.

This may be in the right ballpark; there may be some effect from the intrusion of the mixing polys in each output mixing level.

>Once I've found it, I have a lot of
>information about the four permutations which formed my four
>critical bytes. Repeating the process to find all the matches over
>all 32 permutions, I should be able to break the initial permutations.

I'd like to see a discussion of how one would continue from there to actually find substitution values and start attacking one of the internal DES operations.

I don't see how to recover the substitution element values, but I don't stand on this as a redeeming strength.

>Looking at what happens in the output, I can probably infer a lot
>about the final permutations too.
>
>That's as far as I'll take the attack for now. Perhaps I've
>misunderstood something about the design; please let me know if any
>of the steps I've described is impossible. I'll elaborate if parts of
>my explanation are unclear. If as much as I've described can be done,
>I certainly think it's a strong enough toe-hold for the analyst that
>Fenced DES is dead.

Not at all. Boy, are you looking for a cheap victory :-). You have described an interactive chosen-plaintext attack. This is hardly the same thing as a known-plaintext or ciphertext-only attack.

This attack essentially shows the need for additional system-design requirements. Few of us expect to be able to take an arbitrary cipher, drop it into an arbitrary application and produce a secure system. Instead, the entire system must be designed to properly use the cipher and avoid its weaknesses. This is normal practice.

This attack indicates that any system using this Fenced DES design must prevent a chosen-plaintext attack of the necessary size. This

can be done by not allowing The Opponent to perform chosen-plaintext ciphering, or by forcing a key change before the necessary amount of data has been ciphered. Thus, your adaptive chosen-plaintext attack is itself easily rendered ineffective.

Obviously, the attack has been on the Fencing and Mixing structure, and has not broken DES with this small effort. The guarantee that Fenced DES could not possibly be weaker than DES remains unbroken.

The attack also directly indicates a solution, and that is to mix repeatedly until each and every mixing output byte is a function of each and every mixing input byte. This would require three more (linear!) mixing levels on each side, and would probably *still* be faster than Triple-DES. The mixing itself would be similar to my mixing ciphers which use an internal array of keyed substitutions instead of ciphers.

One of the fundamental questions in a Fencing and Mixing cipher is whether or not linear mixing can be a major part of a strong cipher. This attack has not answered that question.

To absolutely "kill" any particular Fenced DES design, one would have to show that the Fencing and Mixing structure has not improved DES strength at all. To do this, one would need to find a (known-plaintext or ciphertext-only) attack which has a total complexity of 56 or perhaps 58 bits.

To show that a particular Fenced DES design is probably not worth using in the proposed applications, one would need to find a (known-plaintext or ciphertext-only) attack which has a total complexity of under 120 bits.

>Back to the thread...
>
>> Are you convinced about the strength of DES?
>
>Absolutely, positively not. I'm convinced of its weakness, and
>I don't think it should be used as a component in a larger cipher.

Very amusing.

The point is that, for a lot of business people who are responsible for ciphering (despite not having a deep background for it), the only cipher "known" to be secure (at a certain level) *is* DES.

>> >One of the basic
>> >principles in block-cipher design is that each key bit and
>> >data bit should interact with the others many times in
>> >internal transformations.
>>
>> No, this is not strictly true.
>>
>> The principle you are looking for is called "avalanche," and states
>> a *goal* that every output bit should be "affected" by every input
>> bit. (This is no more than a consequence of the observation that
>> a block cipher is a machine which attempts to emulate a Simple
>> Substitution table of impractical size. Every Simple Substitution
>> will avalanche.)
>>
>> The testable result is that a change on any input bit should change
>> (almost) half of the output bits, on average. Any implication that
>> one can only get such a result if bits *explicitly* interact "many
>> times" is not warranted, as far as I know.
>
>No, the criteria I meant was the one I said. Avalanche is not enough
>(and to be fair neither is my criteria; they're both necessary but not
>sufficient). All the bits need to be effected in a complex way. If
>sections of the key are used just once, it often allows the analyst to
>partition the effect of those bits from the effects of others. That
>is why meet-in-the-middle works, why your NxM DES failed, and how I
>attacked Fenced DES.

To be fair, the reason I put in the outline of avalanche reasoning was our previous discussion, where you did not appear to accept that avalanche was inherent in Simple Substitution. You appear to have come some way toward the position which I took, and you opposed, just several years ago.

It may be that your "rule of thumb" is stated too generally to apply directly to new ciphering structures. In these Fencing ciphers, I would say that each keyed substitution has a "full" amount of key. That is, each 8-bit substitution table has a keyed state of 256 bytes for 256! possibilities; finding two keys which produce same state in some substitution table is pretty unlikely. So, changing the cipher key could be expected to change each and every keyed substitution, so that data which encounters even one substitution indeed does interact with the "full" key.

As for why NxM DES failed, one necessary (but insufficient) reason is that it was *proposed*. Failing to propose a cipher may assure lack of failure, but also has no chance of producing a successful design. I consider "design failure" an underrated commodity, because it often takes many such failures to understand the range

of design and achieve even one true success. If failures in some
of my designs highlight new rules for future designs, I claim
progress (if not success), and certainly not personal failure.

In this particular case, I don't consider a defined-plaintext
attack to be any kind of a death-knell for the design. It does
present an operational limitation which must be taken into
account when that particular cipher is designed into a system.

>> The use of keyed invertible substitutions guarantees that *any*
>> change in a substitution input value *will* avalanche the output
>> from the substitution. (The use of keyed substitutions also
>> protects against Differential and Linear attacks, since the
>> transformation is keyed and not known externally.) Again, Feistel
>> operations have no guarantees.

Here we have the guarantees. Avalanche stands guaranteed as stated.
The present attack is a chosen-plaintext Differential attack, but
to the extent that the key is changed often enough, no such attack
is possible.

Actually, chosen plaintext attacks can be prevented by the rest
of the system in a variety of ways, including simply not allowing
arbitrary ciphering access.

Still, it is very important to know the limitations of any cipher,
and I am very glad that you took the time to look at it.

>> The separate use of balanced, complete mixing layers and a "fencing"
>> layer of keyed, invertible substitutions guarantees avalanche
>> across large blocks. The use of another set of layers protects
>> against various "meet in the middle" or "fix in the middle"
>> attacks (and may not always be necessary). The use of a fencing
>> layer before linear mixing protects the mixing, provided that
>> the subsequent ciphering layers have not been exposed.
>>
>I'll let you assess my attack before I argue the problems with that.

The above was the short outline of the basis for strength logic.
While I am certainly grateful for your consideration, I think we'd
need a known-plaintext or ciphertext-only attack to do serious
damage to the concept of ciphering with such a structure.

>> >Finally, I don't like designs which work extensively on
>> >isolated sections of the key and/or data space.
>>
>> Then you should *love* my ciphers.
>
>Uh...
>
>[...]
>> I can provide much *better* guarantees of key interaction than
>> anything I've seen on the DES design.
>>
>
>Obviously I disagree.

Again, I have a serious problem with the phrasing:

"designs which work extensively on isolated sections
of the key and/or data space,"

although I do have a better understanding, now, of what you
probably mean by this.

As you say, I have proposed many ciphers -- whole new classes
of ciphers in fact. Now, how do I apply your rule of thumb to
these many divergent designs?

Consider the 4x cipher which consists of 5 levels of "FFT" mixing
using a keyed pair of orthogonal Latin squares (64KB each).
Now, does each data byte "use" the key repeatedly in any real
sense? Suppose I use a different, separately-keyed, oLs mixing
pair at each mixing junction: Does each data byte *then* use the
key repeatedly? How does one apply your admonition?

>[...]
>> As a matter of fact, this common design concept "let's just make
>> something pretty much like DES, only a little better" is very
>> seriously flawed due to a lack of known mathematical basis for
>> Feistel-type designs. The fact that many do this does not make it
>> a good design technique.
>
>Which side are you arguing? You're the one trying to use DES but add
>some simple layers to make it stronger.

I am using a Feistel design as a *building-block* in a hopefully
better cipher. DES uses S-boxes which are laughably-weak on their
own, but that does not mean that the resulting cipher is quite
so laughable.

DES is what many people consider to be the most proven cipher in the world, and now has problems with respect to keyspace and block size. The obvious alternative of Triple-DES is great for users (who have lots of CPU cycles to spare), but not so great for servers (which have no free cycles). The situation begs the question as to whether it is possible to do better with what we have. (Especially since the academic community has come so late to the party of providing a widely-accepted replacement.)

Note that I have *not* proposed some new Feistel cipher with larger blocks, or Feistel structure using 1x DES as an S-box. I claim that the design of secure S-boxes for a Feistel design is far too tricky, and we know far too little about it. In fact, using a cipher (an invertible function) as a Feistel S-box (instead of some non-invertible function) would make me nervous.

So you tell me, which "side" am I on?

>> Your so-called "established principles" do not, alas, begin to
>> provide the mathematical guarantees of operation provided by
>> Fencing and Mixing cipher designs. It seems strange you would
>> bring this up, for it is the conventional wisdom of Feistel
>> ciphering which is weak here, not these designs.
>>
>> Terry Ritter ritter@io.com http://www.io.com/~ritter/
>
>I think I probably have a few numbers wrong in my analysis, but
>not so far wrong as invalidate the attack on Fenced DES. If
>the attack doesn't work, I'd like to know why.

I'd like some time to think about it, but I think it probably works, as far as it goes.

>If it does,
>well I have to respect you for showing your failures on your web
>page.

I really hope that this was not intended to be as sarcastic as some might take it. Documenting success *and* failure is part of responsible design.

>--Bryan

Thank you! You have my sincere thanks for your analysis, and thanks as well for finding a bug.

While I am naturally disappointed that Fenced DES was not immune to interactive defined-plaintext attack, I do not yet consider the design a "failure." The attack does set some additional system requirements on the use of Fenced DES, and this information is very welcome. It also directly implies how to make structures which are not vulnerable to this attack. The attack also does not address fundamental questions like whether weak linear mixing automatically indicates a weak cipher.

Thanks again.

Terry Ritter ritter@io.com http://www.io.com/~ritter/

========
Path:
news.io.com!arlut.utexas.edu!geraldo.cc.utexas.edu!cs.utexas.edu!howland.erols.net!cam-news-hub1.bbnplanet.com!news.mathworks.com!newsfeed.internetmci.com!newsxfer2.itd.umich.edu!uunet!in3.uu.net!trellis.wwnet.com!news
From: rscott@wwnet.com (Robert Scott)
Newsgroups: sci.crypt
Subject: Re: Ciphers with *Huge* Block Size ?
Date: Wed, 28 Aug 1996 12:45:14 GMT
Organization: WWNET
Lines: 64
Message-ID: <50135n$6i2@trellis.wwnet.com>
References: <321DE177.2193@io.com> <321E0AA1.4517@jhuapl.edu>
<4vmcm7$8jd@nntp-1.io.com> <3220FFAD.27BB@jhuapl.edu> <4vvdbj$3de@anarchy.io.com>
NNTP-Posting-Host: annas112.wwnet.com
X-Newsreader: Forte Free Agent 1.0.82

ritter@io.com (Terry Ritter) wrote:

[..snip..]

> You have
> described an interactive chosen-plaintext attack. This is hardly the
> same thing as a known-plaintext or ciphertext-only attack.

> This attack essentially shows the need for additional system-design
> requirements. Few of us expect to be able to take an arbitrary
> cipher, drop it into an arbitrary application and produce a secure
> system. Instead, the entire system must be designed to properly
> use the cipher and avoid its weaknesses. This is normal practice.

> This attack indicates that any system using this Fenced DES design
> must prevent a chosen-plaintext attack of the necessary size. This

> can be done by not allowing The Opponent to perform chosen-plaintext
> ciphering, or by forcing a key change before the necessary amount of
> data has been ciphered. Thus, your adaptive chosen-plaintext attack
> is itself easily rendered ineffective.

[..snip..]

> I think we'd
> need a known-plaintext or ciphertext-only attack to do serious
> damage to the concept of ciphering with such a structure.

[..snip..]

> While I am naturally disappointed that Fenced DES was not immune to
> interactive defined-plaintext attack, I do not yet consider the
> design a "failure." The attack does set some additional system
> requirements on the use of Fenced DES, and this information is very
> welcome. It also directly implies how to make structures which are
> not vulnerable to this attack. The attack also does not address
> fundamental questions like whether weak linear mixing automatically
> indicates a weak cipher.

It seems that the standard of performance for ciphers has been going steadily up. Long ago ciphers were based on entirely secret designs, which if known, would compromise the cipher. Then we moved to open designs with secret keys, but considered just ciphertext-only attacks. Then as ciphers became better and better, we began asking for resistance to known-plaintext attack, then chosen-plaintext attack. Perhaps in practice, a small quantity of chosen-plaintext is all that an adversary can really obtain. But accademically, it became desireable to ask for ciphers that are resistant to massive chosen-plaintext attack, perhaps even interactively. As impractical as these attacks seem, the fact that they are theoretically possible is going to make a cipher unacceptable for adoption as a standard. (Of course, if the weakness is discovered after adoption, then the investment in the standard may outweigh the purely theoretical weakness.) As I see it, your Fenced DES design is still in the proposed stage. Even if it is true that the fencing adds effectively to the keyspace of the total cipher, and even if it is measureably better than DES is every way, this theoretical weakness will have to be dealt with. You will notice that my NEWDES cipher as described in Bruce's book is dismissed only because of this same theoretical weakness, even though it is has a 120-bit key, is much faster than DES, and has no other weakness.

-Bob Scott

========
Path: news.io.com!usenet
From: ritter@io.com (Terry Ritter)
Newsgroups: sci.crypt
Subject: Re: Ciphers with *Huge* Block Size ?
Date: Wed, 28 Aug 1996 18:16:09 GMT
Lines: 45
Message-ID: <5022bk$kqm@nntp-1.io.com>
References: <321DE177.2193@io.com> <321E0AA1.4517@jhuapl.edu>
<4vmcm7$8jd@nntp-1.io.com> <3220FFAD.27BB@jhuapl.edu> <4vvdbj$3de@anarchy.io.com>
<50135n$6i2@trellis.wwnet.com>
NNTP-Posting-Host: dialup-01-102.austin.io.com
X-Newsreader: Forte Free Agent 1.0.82

In <50135n$6i2@trellis.wwnet.com> rscott@wwnet.com (Robert Scott)
wrote:

> As I see it, your Fenced DES design is still in the
>proposed stage. Even if it is true that the fencing adds effectively
>to the keyspace of the total cipher, and even if it is measureably
>better than DES is every way, this theoretical weakness will have to
>be dealt with. You will notice that my NEWDES cipher as described in
>Bruce's book is dismissed only because of this same theoretical
>weakness, even though it is has a 120-bit key, is much faster than
>DES, and has no other weakness.

Actually, I agree (except that we cannot know that *any* cipher has "no other weakness). But note that even if the current attack is completed, it simply breaks through to the DES level, when then must be attacked on its own. Using a DES layer puts a lower limit on strength for the cipher as a whole, which is an interesting advantage.

My first response is to add two more mixing layers each side, which should solve this particular problem. But I will still have to implement it, if only to see what kind of performance hit it will cause. Very much stronger nonlinear mixings could also be used in the very same structure, but then I would consider the DES overkill (except for the minimum strength guarantee).

As it turns out, I "knew" two different, mutually-contradictory things about Differential attacks, and so did not really understand the problem. It is true, of course, that if we changed the key often enough, no Differential attack could succeed (because the substitutions are keyed), but we also don't want to do that, which means that Differential attacks cannot be rejected out-of-hand simply by using keyed substitutions.

It would be nice if we could find some sort of test which would
certify or even indicate that a design is free of this sort of
Differential attack. Maybe some sort of statistical correlation would
point it out.

> -Bob Scott

Terry Ritter ritter@io.com http://www.io.com/~ritter/

========
Path:
news.io.com!news.thenet.net!newsserver.pixel.kodak.com!bloom-beacon.mit.edu!news-res.gsl.net!news.gsl.net!news.mathworks.com!newsfeed.internetmci.com!in3.uu.net!trellis.wwnet.com!news
From: rscott@wwnet.com (Robert Scott)
Newsgroups: sci.crypt
Subject: Re: Ciphers with *Huge* Block Size ?
Date: Wed, 28 Aug 1996 22:38:04 GMT
Organization: WWNET
Lines: 68
Message-ID: <5025t8$h2p@trellis.wwnet.com>
References: <321DE177.2193@io.com> <321E0AA1.4517@jhuapl.edu>
<4vmcm7$8jd@nntp-1.io.com> <3220FFAD.27BB@jhuapl.edu> <4vvdbj$3de@anarchy.io.com>
<50135n$6i2@trellis.wwnet.com> <5022bk$kqm@nntp-1.io.com>
NNTP-Posting-Host: annas124.wwnet.com
X-Newsreader: Forte Free Agent 1.0.82

ritter@io.com (Terry Ritter) wrote:

>Actually, I agree (except that we cannot know that *any* cipher has
>"no other weakness). But note that even if the current attack is
>completed, it simply breaks through to the DES level, when then must
>be attacked on its own. Using a DES layer puts a lower limit on
>strength for the cipher as a whole, which is an interesting advantage.

True, you have a cipher that is at least as strong as DES. But what
about the cost effectiveness of your additions? By merely relaxing
the requirement that the DES key bytes have redundant parity bits, you
have a cipher that costs no more to implement and has 256 times as big
a keyspace. And perhaps running DES a few more rounds (say 18 or 20
instead of 16) would achieve the same degree of mixing afforded by
your fencing. I don't know it for a fact. But perhaps...
Perhaps the real value of combining different methods into one cipher
as you have done is for insurance. If one component is found to have
a weakness, perhaps the combination will still be strong.

You mentioned before that there may be value to using DES as a
component because it has been so scrutinized and is so widely
accepted. This may be a psycological plus, but the official status of
DES does not impress me very much. In order to place faith in the
process of analysis, you have to see it as essentially a continuous
process. That is, the understanding of the strengths and weakness of
DES will come out little by little as the great brains of our time
focus their energies. But what I think is far more likely is that
progress comes in discrete jumps, some small, some huge. Only when
averaged over time and over many other problems does technological
advance appear uniform (or exponential). But that does not say
anything about progress in analyzing one particular cipher. I would
place a lot more confidence in the progress in general principles
extracted of cryptography. From this point of view, DES is just
another Feistel cipher. Perhaps it is optimum for its size, but it
would be no better than average when compared to Feistel ciphers of
just a slightly larger size (in terms of keyspace or rounds, or block
size). Now although the S-boxes of DES may have been specially chosen
to be strong for their size, I believe that it is quite likely that if
the structure of DES were redesigned with larger randomly chosen
S-boxes, then they would almost certainly be stronger than the ones we
have. (eg. instead of 6 bits of input use 8 bits of input to each
S-box, still using 4 bits of output).

>My first response is to add two more mixing layers each side, which
>should solve this particular problem. But I will still have to
>implement it, if only to see what kind of performance hit it will
>cause. Very much stronger nonlinear mixings could also be used in the
>very same structure, but then I would consider the DES overkill
>(except for the minimum strength guarantee).

It seems to me that the strength of ciphering is based on something
like this. Suppose you have 100 units of resources to spend on
running a cipher. These resource could be time or silicon, but just
think of them as general resources. If you design a cipher composed
of 80 units of one component (say, DES) and 20 units of another
component (say, fencing, or mixings), then the overall "strength" of
the cipher is 80 * 20 = 1600. (Provided the two components don't have
an unfortunate relationship). On the other hand, if the system is
designed with four components, each costing 25 units, then the system
strength is 25 * 25 * 25 * 25 = 390625. If this "product of
resources" analysis holds, it is easy to see that the most
cost-effective cipher would be one that is composed of as many cheap,
non-commuting parts as possible. This looks like just what Feistel
ciphering is doing.

  -Bob Scott

```
========
Path:
news.io.com!news.thenet.net!trellis.wwnet.com!news.inc.net!newspump.sol.net!uwm.edu!math.ohio-state.edu!howland.erols.net!surfnet.nl!tudelft.nl!cyber.tn.tudelft.nl!visser
From: visser@ph.tn.tudelft.nl (Boudewijn W. Ch. Visser)
Newsgroups: sci.crypt
Subject: Re: Ciphers with *Huge* Block Size ?
Date: 28 Aug 1996 20:33:20 GMT
Organization: Delft University of Technology, Faculty of Applied Physics
Lines: 41
Message-ID: <502aeg$93j@cyber.tn.tudelft.nl>
References: <321DE177.2193@io.com> <321E0AA1.4517@jhuapl.edu>
<4vmcm7$8jd@nntp-1.io.com> <3220FFAD.27BB@jhuapl.edu> <4vvdbj$3de@anarchy.io.com>
<50135n$6i2@trellis.wwnet.com>
NNTP-Posting-Host: sextantis.ph.tn.tudelft.nl
X-Newsreader: NN version 6.5.0 (NOV)

rscott@wwnet.com (Robert Scott) writes:

[snip some history]

>for resistance to known-plaintext attack, then chosen-plaintext
>attack.  Perhaps in practice, a small quantity of chosen-plaintext is
>all that an adversary can really obtain.  But accademically, it became
>desireable to ask for ciphers that are resistant to massive
>chosen-plaintext attack, perhaps even interactively.  As impractical
>as these attacks seem, the fact that they are theoretically possible
>is going to make a cipher unacceptable for adoption as a standard.

Since todays cipher may very well end up in a high-bandwith application,
I don't consider such requirements completely "academically".

Many of the historial ciphers were used for very small messages; Both
because of transmission issues (or hand-decoding),but it was realized that
long messages and little  keychanges posed risks.

A modern cipher should also be suitable to encrypt a few gigabytes of data
with no keychange for months or years,and lots of known or even chosen
plaintext.[disk encryption,for example]
Or encrypting over a 100 Mbit network,where an adaptive chosen plaintext
is very well possible.

Besides, if a cipher is vulnerable to a chosen plaintext attack,I would
want a -very- solid proof that it isn't vulnerable to a known-plaintext
attack,if I happened to have a use for it where a chosen plaintext attack
wouldn't be possible. When it comes to trusting ciphers,the motto
is "weak unless there is convincing evidence for the opposite",and that
goes especially for ciphers in which some kind of weakness has already
been found.

Snake-oil sellers work different :"presumed strong until broken".

Boudewijn
--
+-----------------------------------------------------------------+
|Boudewijn Visser       |E-mail:visser@ph.tn.tudelft.nl |finger for |
|Dep. of Applied Physics,Delft University of Technology |PGP-key    |
+-- my own opinions etc ----------------------------------------+


========
Path:
news.io.com!imci4!newsfeed.internetmci.com!cpk-news-hub1.bbnplanet.com!cpk-news-feed1.bbnplanet.com!netnews.jhuapl.edu!usenet
From: Bryan Olson
Newsgroups: sci.crypt
Subject: Re: Ciphers with *Huge* Block Size ?
Date: Wed, 28 Aug 1996 11:09:47 -0400
Organization: Johns Hopkins Applied Physics Lab
Lines: 215
Message-ID: <3224613B.3BEE@jhuapl.edu>
References: <321DE177.2193@io.com> <321E0AA1.4517@jhuapl.edu>
<4vmcm7$8jd@nntp-1.io.com> <3220FFAD.27BB@jhuapl.edu> <4vvdbj$3de@anarchy.io.com>
Reply-To: Bryan_Olson@jhuapl.edu
NNTP-Posting-Host: snake.jhuapl.edu
Mime-Version: 1.0
Content-Type: text/plain; charset=us-ascii
Content-Transfer-Encoding: 7bit
X-Mailer: Mozilla 3.0 (WinNT; I)

I'm thinking of writing up the attack for the research group.
I won't have time for a couple weeks.  There are still a few
open issues.

[Lot's of stuff is edited out at various places.]

Terry Ritter wrote:

>
> In <3220FFAD.27BB@jhuapl.edu> Bryan Olson
> wrote:
> >(To follow the next section, readers will have to understand
> >Ritter's "Fenced DES".  See: http://www.io.com/~ritter/FENCED.HTM
> >)

And to follow this post, readers will have to understand the
```

attack in the referenced post.

> >
> >In your block mixing functions, it seems that a change of one bit
> >position in either half of the input can change only the output bit
> >in the same position and the one to the immediate left (circularly).
> >Am I wrong on that?
>
> Just technically. We have a field; sometimes the field
> polynomial is activated and we get its bits too. Usually not.
>

Right. It looks like the high-order bit of each block triggers
addition of the polynomial.

> >If so, the change between a plaintext A and A'
> >needed to make three of the internal DES applications receive the
> >same input are not all that "substantial". Even with the
> >substitutions we could certainly search for them.
>
> Hardly "certainly." Search implies recognition of difference.
> It would be easy to cycle through all combinations of four bytes
> and know that somewhere in there the combination(s) we wanted
> must exist. The problem is knowing when that occurred.
>

I thought I answered that. The output pattern I described is
what allows recognition.

> >I can pick four bytes that will be mixed only with each other in
> >both levels of pre-DES mixing.

> This doesn't take much "picking"; this is the situation in
> general. [...]

Right. The simple picking criteria is four bytes in the
same relative position in the four eight-byte sections of the
input, and not the most significant bytes of these sections.

> >If A and A' are a match, I
> >can immediately find many other matches by changing bytes which don't
> >interact with the critical four. That gives me a good space to search
> >for any statistical anomaly which would indicate a match.
>
> If we *assume* that A and A' match, we can do lots of things.
> The issue is finding that match.
>

Is this point still open, or do you agree that I can find and
recognize the matches?
>
> You are proposing a situation where particular bit-position values
> occur "at random" from an internal DES operation. The probability
> of this would seem to be 1 in 2**10.
>

Agreed. That's the number I was using.

> >The block mixing will preserve
> >8 of them, lined up with a single substitution at the end. Thus in the
> >output, we'll see four bytes, each three bytes apart, which are
> >identical
> >under A and A'.

> Generally, yes. Three levels of mixing poly will each tend to
> activate about half the time, though.

Good point. I can avoid the polynomials in the pre-DES
mixing by special handling of the high order bytes. In
the   post-DES I can't control these. I only see two
"levels" of mixing, (though three mixes).

In my description "Three bytes apart" should have been
"seven bytes apart".

>
> >If that happens, then all 4 DES outputs will have 10 bits positions
> >which are the same under A as under A'.
>
> Yes. If that happens.
>
> >If they hold under a non-match, then the four-byte coincidence
> >still has the same probability, but if they hold
> >under a match, then the coincidence is a certainty.
>
> ?
>
I was simply pointing out that the 1 in 2**10 shot can
occur whether or not I have a match. It increases the
chances of the visible pattern in the output only if it
holds under a match.

>
> >So I have to search over a 2^32 space, and run a few hundred pairs
> >for each to find such a match.

>
> This may be in the right ballpark; there may be some effect from
> the intrusion of the mixing polys in each output mixing level.
>

Does this drop the chance of the output pattern under
a match by a factor of 4 ? That does make the search
harder. It shouldn't change the over-all complexity
since attacking DES still dominates.

I'm not yet sure whether the chance of polynomial
modulation helps or hurts the attack. I may be able
to find specific bits of output by detecting when
the modulation occurs.

> >Once I've found it, I have a lot of information about the
>
> I'd like to see a discussion of how one would continue from there
> to actually find substitution values and start attacking one of
> the internal DES operations.
>
> I don't see how to recover the substitution element values, but I
> don't stand on this as a redeeming strength.
>

Right. I still need to work out just what I can infer about
the substitutions. Your posts don't specify exactly how the
substitutions are produced from a key, so at some point I'll
have to leave open how I can get from what I know about the
substitutions to how to determine the rest.

You also don't specify a polynomial. I'm not sure it matters.

[...]
> >I certainly think it's a strong enough toe-hold for the analyst that
> >Fenced DES is dead.
>
> Not at all. Boy, are you looking for a cheap victory :-). You have
> described an interactive chosen-plaintext attack. This is hardly the
> same thing as a known-plaintext or ciphertext-only attack.
>

From the FAQ:

A strong encryption algorithm will be unbreakable not only under known
plaintext (assuming the enemy knows all the plaintext for a given
ciphertext) but also under "adaptive chosen plaintext" -- an attack
making life much easier for the cryptanalyst. In this attack, the
enemy
gets to choose what plaintext to use and gets to do this over and
over,
choosing the plaintext for round N+1 only after analyzing the result
of
round N.

>
> Obviously, the attack has been on the Fencing and Mixing structure,
> and has not broken DES with this small effort. The guarantee that
> Fenced DES could not possibly be weaker than DES remains unbroken.
>

Agreed.

> The attack also directly indicates a solution, and that is to mix
> repeatedly until each and every mixing output byte is a function
> of each and every mixing input byte. This would require three more
> (linear!) mixing levels on each side, and would probably *still* be
> faster than Triple-DES. The mixing itself would be similar to my
> mixing ciphers which use an internal array of keyed substitutions
> instead of ciphers.
>

Also from the FAQ:

If you don't have enough
experience, then most likely any experts who look at your system will
be able to find a flaw. If this happens, it's your responsibility to
consider the flaw and learn from it, rather than just add one more
layer of complication and come back for another round.

> To be fair, the reason I put in the outline of avalanche reasoning
> was our previous discussion, where you did not appear to accept
> that avalanche was inherent in Simple Substitution. You appear to
> have come some way toward the position which I took, and you
> opposed, just several years ago.
>

Not at all. I was encouraging you to say something well defined
and true, instead of something ill-defined or false.

> >[...]I have to respect you for showing your failures on your web
> >page.
>
> I really hope that this was not intended to be as sarcastic as

> some might take it. Documenting success *and* failure is part
> of responsible design.

Not meant sarcasticly. I hope quoting the FAQ didn't come
off as too snotty either; I just wanted to make it clear
that I can justify calling the attack successful cryptanalysis
of 4X Fenced DES (provided I can advance it at a few points).

--Bryan

========
Path: news.io.com!usenet
From: ritter@io.com (Terry Ritter)
Newsgroups: sci.crypt
Subject: Re: Ciphers with *Huge* Block Size ?
Date: Wed, 28 Aug 1996 18:16:13 GMT
Lines: 126
Message-ID: <5022bq$kqm@nntp-1.io.com>
References: <321DE177.2193@io.com> <321E0AA1.4517@jhuapl.edu>
<4vmcm7$8jd@nntp-1.io.com> <3220FFAD.27BB@jhuapl.edu> <4vvdbj$3de@anarchy.io.com>
<3224613B.3BEE@jhuapl.edu>
NNTP-Posting-Host: dialup-01-102.austin.io.com
X-Newsreader: Forte Free Agent 1.0.82

In <3224613B.3BEE@jhuapl.edu> Bryan Olson <Bryan_Olson@jhuapl.edu>
wrote:

>> >If A and A' are a match, I
>> >can immediately find many other matches by changing bytes which don't
>> >interact with the critical four. That gives me a good space to search
>> >for any statistical anomaly which would indicate a match.
>>
>> If we *assume* that A and A' match, we can do lots of things.
>> The issue is finding that match.
>>

>Is this point still open, or do you agree that I can find and
>recognize the matches?

These were just comments made while reading the text for the first
time.

>> Obviously, the attack has been on the Fencing and Mixing structure,
>> and has not broken DES with this small effort. The guarantee that
>> Fenced DES could not possibly be weaker than DES remains unbroken.
>>

>Agreed.

>> The attack also directly indicates a solution, and that is to mix
>> repeatedly until each and every mixing output byte is a function
>> of each and every mixing input byte. This would require three more
>> (linear!) mixing levels on each side, and would probably *still* be
>> faster than Triple-DES. The mixing itself would be similar to my
>> mixing ciphers which use an internal array of keyed substitutions
>> instead of ciphers.
>>

>Also from the FAQ:

> If you don't have enough
> experience, then most likely any experts who look at your system will
> be able to find a flaw. If this happens, it's your responsibility to
> consider the flaw and learn from it, rather than just add one more
> layer of complication and come back for another round.

First of all, I am *not* adding a layer of complication, I am
proposing to fix the mixing layer so that it will not have the
weakness you indicate. There still would be the same five-layer macro
structure consisting of Fencing, linear mixing, DES, linear mixing,
and Fencing. This would still be very recognizable as Fenced DES.

Given that there is no scientific *proof* of a secure cipher, the only
way a design can escape a sequence of modifications is by luck, and we
can't teach luck. It might be nice to keep those changes "in house,"
which is what we had with DES, but this requires massive resources and
several different groups of people working on the problem, and can
still only obscure the fact that a sequence of modifications is the
way ciphers are built. And even after extensive S-box analysis
in-house, apparently DES came back from NSA with different S-boxes;
should the designers have thrown it out and started over?

Any really innovative construction is necessarily a *process* rather
than an *event*. Only *after* a structure has been evolved can we
avoid the possiblility of having it fail. It would be nice if the
various ciphering structures had already been investigated and
analyzed and the results placed in a handbook for use, but we seem to
be at least a generation away from that.

In fact, I have no idea what the FAQ could possibly mean except "start
over with a completly new structure" or "if you ever fail, get out of
the business," both of which are incredibly ignorant and arrogant
responses. I note that there is no suggestion that an *analyst* who

failed to break a cipher should get out of the business (since obvioiusly *somebody* knows something the analyst does not, so he has not learned his field). Presumably, analysts are allowed to fail and learn, but failed cipher designers might as well quit. I guess we know which classification wrote this part of the FAQ!

>> To be fair, the reason I put in the outline of avalanche reasoning
>> was our previous discussion, where you did not appear to accept
>> that avalanche was inherent in Simple Substitution. You appear to
>> have come some way toward the position which I took, and you
>> opposed, just several years ago.
>>

>Not at all. I was encouraging you to say something well defined
>and true, instead of something ill-defined or false.

>> >[...]I have to respect you for showing your failures on your web
>> >page.
>>
>> I really hope that this was not intended to be as sarcastic as
>> some might take it. Documenting success *and* failure is part
>> of responsible design.

>Not meant sarcasticly. I hope quoting the FAQ didn't come
>off as too snotty either; I just wanted to make it clear
>that I can justify calling the attack successful cryptanalysis
>of 4X Fenced DES (provided I can advance it at a few points).

I have several points here; the first is that the situation is not black or white: we can judge one cipher better than another without saying that the inferior is useless. Similarly, we can also judge attacks, and some are better than others. Currently, ciphers are being held to one heck of a high standard -- which is OK -- but attacks can be pretty casual. Few attacks ever seem to be reduced to practice and actually tested before proposal, but this is an integral part of the construction of ciphers. And if the attack itself sugggests a response, I wonder why the attacker does not have to deal with that as well? The whole process seems biased away from that which would best end up producing a sollid improved design, and toward some sort of ego contest which I reject.

That said, I am very happy for Bryan to have found any sort of weakness in one of my designs, and am willing to call it "broken" on that account and move on to prevent that sort of attack. I am probably the last person to judge such an attack on its overall qualities, because if even *part* of the attack can do something I did not think could be done, it has already triggered a redesign.

>--Bryan

Terry Ritter ritter@io.com http://www.io.com/~ritter/

========
Path:
news.io.com!insync!apollo.isisnet.com!eru.mt.luth.se!news.kth.se!nntp.uio.no!Norway.EU.net!EU.net!nntp04.primenet.com!nntp.primenet.com!howland.erols.net!newsfeed.internetmci.com!newsxfer2.itd.umich.edu!caen!umass.edu!kernighan.cs.umass.edu!usenet
From: Lewis McCarthy <lmccarth@cs.umass.edu>
Newsgroups: sci.crypt
Subject: Re: Ciphers with *Huge* Block Size ?
Date: Mon, 26 Aug 1996 14:12:41 -0400
Organization: Theoretical Computer Science Group, UMass-Amherst
Lines: 31
Message-ID: <3221E919.2781@cs.umass.edu>
References: <321DE177.2193@io.com> <321E0AA1.4517@jhuapl.edu>
<4vmcm7$8jd@nntp-1.io.com>
NNTP-Posting-Host: opine.cs.umass.edu
Mime-Version: 1.0
Content-Type: text/plain; charset=us-ascii
Content-Transfer-Encoding: 7bit
X-Mailer: Mozilla 3.0 (X11; I; OSF1 V3.0 alpha)

Terry Ritter writes:
> No. Modern mixing guarantees distribution of any input change to
> every substitution in the next layer in n log n operations.
> (This is either log n layers of n-byte multi-precision mixing, or
> log n layers of byte-by-byte mixing.) Feistel mixing gives no such
> guarantees, and, indeed, virtually requires experiment to decide
> how many rounds are "enough." (Authors of similar Feistel ciphers
> often let the user decide how many rounds are enough, as though
> users will have the background to do this.)

End users already need to pick key sizes to achieve a desired balance between speed and cryptographic security. Ideally a grounding in the contemporary state-of-the-art of cryptanalysis and computational feasibility informs the selection. This doesn't prevent inexpert users from making good choices about their keying material in presently deployed cryptosystems. The cryptologic community can and does provide both customized and canned advice on this and related issues to everyone else.

I'm not claiming that end users of round-based ciphers should be choosing the number of rounds they use. We need to be careful about what we mean by "the user of a cipher". I believe it is reasonable to

expect an implementor to decide upon a range or fixed number
of rounds of a cipher (hash fn., etc.) when integrating cryptography
into an application. This is just one of a host of concerns that
should be addressed to ensure that an appropriate cryptosystem is
properly deployed to satisfy a specified security requirement.
--
Lewis http://www.cs.umass.edu/~lmccarth/lmkey.asc
"He said, `You are as constant as the Northern Star,' and I said,
`constantly in the dark ? where's that at ?'" -- Joni Mitchell

========
Path:
news.io.com!arlut.utexas.edu!cs.utexas.edu!math.ohio-state.edu!uwm.edu!newsfeed.internetmci.com!in2.uu.net!munnari.OZ.AU!harbinger.cc.monash.edu.au!newsroom.utas.edu.au!mg4-71.its.utas.edu.au!user
From: roger_sf@postoffice.utas.edu.au (Roger Fleming)
Newsgroups: sci.crypt
Subject: Re: Ciphers with *Huge* Block Size ?
Followup-To: sci.crypt
Date: 24 Aug 1996 08:20:10 GMT
Organization: University of Tasmania
Lines: 37
Message-ID: <roger_sf-240896173928@mg4-71.its.utas.edu.au>
References: <321DE177.2193@io.com>
NNTP-Posting-Host: mg4-71.its.utas.edu.au

Terry Ritter <ritter@io.com> wrote:

[...]
> Next, it is clear that codebook attacks can start to be effective
> when we get two ciphertext blocks which are the same. If we assume
> an absolutely flat plaintext distribution (we assume cipher block
> chain (CBC) mode, or other plaintext randomizer) and 64-bit blocks,
> we can expect to see this with about 2**32 blocks, and rapidly
[...]
> In fact, the whole need to use CBC (or other plaintext randomization)
> is based on the small amount of language entropy in 8 character
[...]
>
> On the other hand, a 200-byte block of plaintext language should
> have about 300 bits of entropy, so we could expect to find two the
> same in about 2**150 blocks.[...]

Codebook attacks are a serious threat on low entropy texts, but chaining
solves the problem completely and efficiently, and certain other problems
as well.

I agree wholeheartedly that even for high entropy texts, 64 bit wide blocks
will one day (real soon now) reach the end of their useful lives. But that
doesn't mean we have to go all the way to 1600 bit blocks. Just as with key
sizes, quite a small increment should be good for a very long time indeed.
We don't require (and probably never will) 1600 bit symmetric keys to
defeat keysearch attacks; and we don't require 1600 bit blocks to defeat
codebook attacks.

Consider a 128 bit block cipher. Quite apart from possibly anticipating 64
bit processors (or new UFN designs), it should take about 2**64 blocks, or
262,144 exabytes of data before codebooks start to be any use at all (if
used in a chaining mode). This is equivalent to about 1620 channels of real
time hi-res 24-bit colour video encrypted continuously for 100 years. The
storage system for this codebook, once completed, will require billions of
tonnes of matter (even assuming storage systems requiring only one atom per
bit).

========
Path: news.io.com!usenet
From: ritter@io.com (Terry Ritter)
Newsgroups: sci.crypt
Subject: Re: Ciphers with *Huge* Block Size ?
Date: Sun, 25 Aug 1996 07:25:19 GMT
Lines: 89
Message-ID: <4vovcb$ap5@nntp-1.io.com>
References: <321DE177.2193@io.com> <roger_sf-240896173928@mg4-71.its.utas.edu.au>
NNTP-Posting-Host: dialup-01-153.austin.io.com
X-Newsreader: Forte Free Agent 1.0.82

In <roger_sf-240896173928@mg4-71.its.utas.edu.au>
roger_sf@postoffice.utas.edu.au (Roger Fleming) writes:

>Terry Ritter wrote:
>
>[...]
>Codebook attacks are a serious threat on low entropy texts, but
>chaining
>solves the problem completely and efficiently, and certain other
>problems
>as well.

So much for the positive side, but chaining has significant
costs and actually *creates* certain problems. If we can avoid
chaining, we may simplify things in at least some applications.

>I agree wholeheartedly that even for high entropy texts, 64 bit

>wide blocks
>will one day (real soon now) reach the end of their useful lives.
>But that
>doesn't mean we have to go all the way to 1600 bit blocks. Just
>as with key
>sizes, quite a small increment should be good for a very long
>time indeed.
>We don't require (and probably never will) 1600 bit symmetric
>keys to
>defeat keysearch attacks; and we don't require 1600 bit blocks
>to defeat
>codebook attacks.

I believe I generally agree with the motivation behind this.
That is, dramatically larger blocks or keys are not necessary
for strength. Real cryptographic strength should be measured as
the cheapest way to gain the hidden information, and few of us
could possibly maintain physical security sufficient to prevent
billion-dollar intrusions (or even thousand dollar intimidations).

Nevertheless, from the design standpoint, once one has a flexible
efficient, large-block technology, *why not* use large blocks and
keys? Surely it is obvious that an efficient keying mechanism
should handle short keys very well -- then if one wants to use keys
with only 96 bits, that's fine. But the basic technology could
support real keyspaces in excess of 1000 bits with the exact same
efficient design. (The storage of 1000-bit keys might have once
been a problem, but those days are gone.) The situation is similar
with large blocks. The added costs for larger keys or blocks are
not linear, especially with new technology.

There *is* an argument, though, for 200-byte blocks (or so!), which
is the expected language entropy (~300 bits) and the resulting
level for codebook attacks (2**150 blocks). We might want this
last figure to be 2**120 or more to take this particular issue off
the table, and that implies 160-byte blocks. Again, a
slightly-larger block is not an implementation hardship or cost.

>Consider a 128 bit block cipher. Quite apart from possibly
>anticipating 64
>bit processors (or new UFN designs), it should take about 2**64
>blocks, or
>262,144 exabytes of data before codebooks start to be any use
>at all (if
>used in a chaining mode).

Fine, but we can *avoid* the use of chaining if we use substantially
larger blocks. Avoiding chaining better supports:

1) use in packet-switching environments (packets are often
delivered out of sequence),
2) random disk access, and
3) parallel processing (as has recently come to my attention :-).

>This is equivalent to about 1620 channels of real
>time hi-res 24-bit colour video encrypted continuously for 100
>years. The
>storage system for this codebook, once completed, will require
>billions of
>tonnes of matter (even assuming storage systems requiring only
>one atom per
>bit).

Terry Ritter ritter@io.com http://www.io.com/~ritter/

========
Path:
news.io.com!insync!news.ios.com!news2.cais.net!news.cais.net!tezcat!cam-news-hub1.bbnplanet.com!news.mathworks.com!newsfeed.internetmci.com!in3.uu.net!info.htcomp.net!NewsWatcher!user
From: wtshaw@htcomp.net (W T Shaw)
Newsgroups: sci.crypt
Subject: Re: Ciphers with *Huge* Block Size ?
Date: 23 Aug 1996 18:36:09 GMT
Organization: Another Netscape News Server User
Lines: 50
Message-ID: <wtshaw-2308961339080001@207.17.188.113>
References: <4vi4pb$fsq@elna.ethz.ch>
NNTP-Posting-Host: 207.17.188.113

In article <4vi4pb$fsq@elna.ethz.ch>, gec@acm.org wrote:

> Hello everybody,
>
> current data encryption techniques usually encrypt a data stream or
> small blocks of data. While this makes perfect sense for communication
> where you need to encrypt and decrypt 'on the fly', it is perhaps not
> the optimal solution for archiving, and generally not the strongest way
> to secure data. These are just idle thoughts, and I would be quite
> interested to hear what you think about it.

You are absolutely right. Given the same algorithm, the larger the block,
the more possibilities for keys, fewer blocks are used, and it is apt to

beharder to attack by brute force. Keeping things in nice small blocks may
appear to make it easier to process, but that is no real excuse.
>
> Imagine a data file (or semantic entity, or whatever) being encrypted
> as a whole, which means that each of the output bits depends on each of
> the input bits (and naturally the secret key). This way, many kinds of
> attacks may be a bit less feasible e.g. imagine doing a known plaintext
> attack on something which simply can not be known in its entity, but of
> which certain parts are well known (e.g. contents of encapsulated IP
> headers, or headers of archived and encrypted mail) -- or imagine doing
> a linear CA attack when the relevant block size is about a Megabyte.
>
> Current block cipher chaining methods (CBC, OFB, CFB, ...) do not offer
> this property, the best result is that the last encrypted 'block'
> depends on all previous elements.

The down side is that when parts depend on other parts, then the message
is increasingly subject to random errors completely destroying the ability
to decode the message.

A good compromise of all is to use longer blocks up to an fairly large
size as a limit. You would have some messages below that limit, others
using several blocks. You also have to solve the varying size problem of
blocks. If you have no limit to size of the possible blocks, you create
another problem, one of defining keyspace.

One class of algorithms I use breaks messages into blocks that might
approach 250 characters in length for some blocks, a good common maximum
for string sizes for most higher level programming languages. Having the
longer strings as a floor for manipulation allows me to get beyond the
popular tinkertoy, bit-oriented algorithms.
/\
wtshaw@htcomp.net Mac Crypto Programs
 You should at least know how to use ROT13.
"Fhpprff vf n Wbhearl, Abg n Qrfgvangvba."
          http://www.htcomp.net/wts/
\/

========
Path: news.io.com!usenet
From: Terry Ritter
Newsgroups: sci.crypt
Subject: Re: Ciphers with *Huge* Block Size ?
Date: Fri, 23 Aug 1996 23:34:25 -0500
Organization: Ritter Software Engineering
Lines: 259
Message-ID: <321E8651.1521@io.com>
Reply-To: ritter@io.com
NNTP-Posting-Host: dialup-01-145.austin.io.com
Mime-Version: 1.0
Content-Type: text/plain; charset=us-ascii
Content-Transfer-Encoding: 7bit
X-Mailer: Mozilla 3.0 (Win95; I)
CC: Jerry Leichter <leichter@smarts.com>

Jerry Leichter <leichter@smarts.com> wrote

> Terry Ritter wrote:
> > [Interesting examples of uses for ciphers with large, or even
> > variable, block sizes.]
>
> The counter-arguments that have held sway are more
> application-oriented
> and historical than theoretical:
>
>   1. Hardware implementations of large block sizes are more
>   difficult.

 Whenever one does something new, it will have new costs, new
 tradeoffs, and new ways to do things. Certainly it is easier to
 build a chip with a 64 data I/O pins than one with 1600 I/O pins;
 therefore, don't do that. Bus bandwidth limitations are problems
 shared with video displays, and could have similar solutions.

 As an erstwhile chip architect myself, I would say that *both*
 Fenced DES *and* Variable Size Block Cipher technologies have
 very attractive hardware implementations, even if they will never
 be quite as straightforward as DES. On the other hand, I note
 that the most successful CPU chips are *not* built on the idea
 that "this chip was easy to implement." Indeed, we often have
 the opposite situation entirely.

>   2. The biggest demand is for encryption of communication
>   streams.

 I'm a little confused here: This discussion so far has been about
 block ciphers. Certainly stream ciphers are important, and I have
 addressed them in other work, but here we discuss *block* ciphers.

>   This adds a number of constraints. For example, stream
>   mode operation is highly desireable; you can't afford
>   to buffer large amounts of information just for
>   encryption.

I guess the issue here is the phrase "large amounts of information." Inevitably, disk sectors *are* written as buffered blocks, and generally are 512 bytes or larger. This amount of data is *already buffered*. Similarly, most modern communications work on "packets" or "frames," which are also *already buffered*. These natural buffer sizes are far, far larger than 8-byte DES blocks.

Note that the main reason we use cipher block chain (CBC) mode with DES ("stream mode"?) is because of the disastrously small amount of language entropy in an 8-character block. In contrast, it is much less necessary to chain large blocks -- we can use electronic code book (ECB) mode and simply cipher the blocks. This is a big advantage *for* random disk access or communications systems (like IP) which may deliver packets out of order.

>(The original question in this thread
> concerned encryption modes in which the output depended
> on the *entire* input, no matter how large. Such a
> mode would be entirely unusable in a communications
> framework.)

Almost anything, taken to excess, is impractical or dangerous.

Since most messages are "short" (relative to available store), they are often completely buffered anyway. These can be ciphered as single units, and larger messages ciphered as a sequence of blocks which are *much* larger than 8-byte DES. There is no need to go overboard on this, since huge benefits can be reaped from relatively small buffers.

> 3. The larger the block, the larger the average space that
> is wasted to fill out short blocks. In a communications
> framework, many messages are quite short. A really
> large block could end up devoting more space to fill
> than to useful message!

I note that, in this case, it would be very nice to have a Variable Size Block and not *have* wasted space, in general.

But even a Fencing or Mixing cipher can be one of a *set* of ciphers of various widths. We use the large one as long as we can fill the block, then use smaller and smaller block sizes at most once each at the end of the message. The remaining "wasted space" would be that of the smallest block implementation, and comparable to that in DES.

> 4. All other things being equal, a large-block cipher is
> probably going to be slower than a small-block cipher
> on a per-bit basis: If each bit of the output depends
> on each bit of the input, then the minimum possible
> operations per bit doubles when you double the block
> size.

This argument is flat-out false, because modern mixing operations (which assure that every output bit depends upon every input bit) can be much faster than Feistel ciphering, which depends upon many repeated rounds to get a good mixing. Even my first Fenced DES implementation would cipher a 4x block at data rates much faster than Triple DES.

Now, if we use DES as a component, we clearly cannot go faster than DES, but we can *approach* DES speeds in a much wider and stronger cipher. Fencing and Mixing technology which does not use DES can have better throughput.

>(This kind of argument is necessarily vague and
> incomplete, since with a larger block size you may be
> able to get away with, say, fewer rounds for equivalent
> security.

If you want better technology, the whole issue of Feistel ciphering with its implicit repeated "rounds" of operation has got to go. By combining mixing and ciphering in one operation, Feistel ciphering does neither optimally well. We can improve on both functions by performing them separately.

>But I know of no ciphers that actually come
> out ahead for large block sizes on this basis.)

Both the Fencing and Mixing and Variable Size Block Cipher technologies provide better mixing performance and so need fewer layers. Unlike Feistel technology, none of these layers are repeated in "rounds."

We now have about 20 years of optimization experience with DES, and just a couple of early implementations of VSBC's. I expect a mature VSBC design to offer wider, dynamically-variable size blocks, much greater strength, inherent resistance to Differential and Linear cryptanalysis, with performance comparable to or better than the best DES implementations, on a byte-per-byte basis.

> 5. Many of these issues grow less important with each passing
> year as chips get faster

No matter how fast a chip is, it can never be as fast as our dreams. Indeed, the more our dreams are realized, the more advanced they get.

(though encryption of traffic
> on the highest speed lines remains problematical -
> transmission data rates are going up faster than CPU
> speeds these days, and the trend lines indicate that
> this will continue. If it does, we'll have to go for
> parallelism of some sort - perhaps just pipelining is
> enough - and that's much easier to do with smaller
> block sizes.)

Please. While parallelism probably *would* be difficult with multi-megabyte blocks, let's be practical. Frankly, ciphering blocks of a few hundred bytes or a few KB is likely to be *more* efficient than ciphering blocks of 8 bytes each. And if we use a large block size, we probably don't need to chain the blocks, but can instead cipher them independently, which is a *tremendous* advantage *for* parallelism.

When we have a block-size choice, we can choose what we want for best operation; if we have no choice, we can only take what we get. Which is likely to produce the better product?

>However, there is an existing infra-
> structure now. Block sizes are often "frozen in" to
> existing designs and difficult and expensive to change.
> That's one reason people like triple-DES: It can be
> used as a drop-in replacement for DES, with effects
> only on the relatively low-frequency activity of key
> distribution.

Throughput (or lack of it) is one reason that some systems managers absolutely *hate* Triple-DES. While individual users have CPU cycles to burn, servers get faster and faster yet end up farther behind. The growth in numbers of users, and the simultaneous growth in the desires of those users, far outpace our ability to scale-up current systems with new hardware. Fundamentally new system structures are often required.

Fenced DES can be highly compatible with DES: we use 4x blocks up to the end of a message, then use 2x and 1x ciphering, thus producing no expansion over ciphering with DES alone.

Clearly, even a VSBC can be made to cipher in blocks of size mod 8.

> 6. Ritter's arguments are strongest when applied to file (or
> disk) encryption, and larger block sizes might be very
> appropriate there (up to some limit, since you don't
> want the cipher block size to force you to transfer
> significantly more data to and from the disk).

I see no reason why any particular technology must necessarily be best at every possible job. But were one single cipher to be picked, an efficient cipher with dynamically variable size blocks (which can be huge blocks or DES blocks), would seem very attractive

>However,
> there is also a strong pressure to standardize on one
> encryption technology - DES today, who-knows-what
> tomorrow.

This is a serious fundamental error. Instead of standardizing on any particular *cipher*, we should innovate a standard *interface* and the ability to plug-in "any" desired cipher. Then, if any particular cipher is found to have problems, any of us using that cipher could switch to another. Note that in this environment, there may be relatively few users using any particular cipher.

(Both ends need the same cipher, of course, but this can be automatically negotiated in the background. Everybody could be expected to have some common ciphers as a part of their basic package, plus various custom ciphers which could even be distributed as a part of Public Key delivery.)

Compare the ability to plug in new ciphers to the current situation where we know that DES is obsolete and yet are coerced by our investment into using it or something very much like it.

At one time, the idea of selecting from among arbitrary ciphers was being seriously considered in the ANSI banking standards committee X9F3.

>If one decides that it's a good idea to use
> the same cryptosystem for both files and communications
> (which has its pluses, like the ability to use fast
> encryption chips developed for communications in disk
> controllers), then constraints on the communications
> side "leak through" to the file side, even if they

```
>               aren't really relevant there.

 If one must standardize on one particular cipher for many different
 applications, a Variable Size Block Cipher has got to be attractive.

>               (Of course, from a security point of view, the more
>               strong cryptosystems you have, the better.

Indeed.  See my web pages:      http://www.io.com/~ritter/


>All the
>               available evidence indicates that NSA produces many
>               different cryptosystems for different purposes.  That's
>               just prudent design.  The same is actually true for any
>               component with security implications - a security hole
>               in, say, Netscape implies a vulnerability on the part
>               of almost all Web users these days.  Nevertheless, the
>               pressures for standardization are so great that they are
>               often impossible to resist.)

 Standards come and go, and, nowadays, we could field a completely
 new ciphering standard in a year or so.  Look at how fast HTML has
 changed.

>
>                                                       -- Jerry
>
 ---
 Terry Ritter   ritter@io.com   http://www.io.com/~ritter/


========
Path:
news.io.com!insync!news.ios.com!news2.cais.net!news.cais.net!tezcat!cam-news-hub1.bbnplanet.com!news.mathworks.com!newsfeed.internetmci.com!news.sgi.com!mr.net!winternet.com!schneier
From: schneier@counterpane.com (Bruce Schneier)
Newsgroups: sci.crypt
Subject: Re: Ciphers with *Huge* Block Size ?
Date: Sat, 24 Aug 1996 16:53:24 -0500
Organization: Counterpane Systems
Lines: 31
Message-ID: <schneier-2408961653240001@news.winternet.com>
References: <4vi4pb$fsq@elna.ethz.ch>
NNTP-Posting-Host: ppp-66-40.dialup.winternet.com
X-Newsreader: Yet Another NewsWatcher 2.2.0b7

There has not been a lot of research on LARGE block sizes, probably
because they are so much harder to analyze.   Block ciphers with small
block sizes are more versitile, and we know all about using feedback
mechanisms to encrypt large blocks of plaintext will small block ciphers.
Still, it makes more than a little sense to encrypt hard drive sectors
with block algorithms that operate on the entire sector as a single block.

There are some serious problems with making a single block cipher with a
block size in the multi-hundred byte range (as opposed to cascading a
block cipher, stream cipher, and block cipher chaining in the other
direction, which will do much the same thing).

1.  It is very hard to get secure diffusion of every plaintext bit to
every ciphertext bit.  There are a lot more possibilities for good
differentials to attack.  On the other hand, you'll probably never get the
volume of plaintext required for a differential attack.

2.  It is a royal pain to analyze large block ciphers.  It is hard enough
to analyze constructions out of smaller primitives.

I kind of like the idea, but I worry about its security.

Bruce

*****************************************************************************
* Bruce Schneier              APPLIED CRYPTOGRAPHY, 2nd EDITION is
* Counterpane Systems         available.  For info on a 15%
* schneier@counterpane.com    discount offer, send me e-mail.
*
* For Blowfish C code, see ftp.ox.ac.uk:/pub/crypto/misc/blowfish.c.gz
*****************************************************************************


========
Newsgroups: sci.crypt
Path:
news.io.com!arlut.utexas.edu!cs.utexas.edu!howland.erols.net!cam-news-hub1.bbnplanet.com!news.mathworks.com!newsfeed.internetmci.com!in3.uu.net!news.dsndata.com!backbone!backbone!wayne
From: wayne@backbone.uucp (Wayne Schlitt)
Subject: Re: Ciphers with *Huge* Block Size ?
In-Reply-To: schneier@counterpane.com's message of Sat, 24 Aug 1996 16: 53:24 -0500
Message-ID: <WAYNE.96Aug24215404@backbone.uucp>
Sender: wayne@backbone (Wayne Schlitt)
Reply-To: wayne@cse.unl.edu
Organization: The Backbone Cabal
References: <4vi4pb$fsq@elna.ethz.ch> <schneier-2408961653240001@news.winternet.com>
Date: Sun, 25 Aug 1996 03:54:04 GMT
Lines: 40

In article <schneier-2408961653240001@news.winternet.com> schneier@counterpane.com
```

(Bruce Schneier) writes:
> There are some serious problems with making a single block cipher with a
> block size in the multi-hundred byte range
>
> 1. It is very hard to get secure diffusion of every plaintext bit to
> every ciphertext bit. [...]

Is it really that important that _every_ plaintext bit gets diffused
to _every_ ciphertext bit? Lets look at what happens when you try to
decrypt a message using a smaller block size with the standard
encryption modes:

```
ECB     each bit only effects every bit in the small block size
CBC     each bit only effects every bit in the small block size plus
        the immediately following block.
CFB     same as CBC
OFB     same as ECB
```

Yes, when you change a single bit in the plaintext and use CBC or CFB,
every following block changes, but the following blocks also have
"known plaintext" in the form of the previous blocks ciphertext.
Also, neither CBC nor CFB change the preceding blocks of ciphertext,
so you know which block contains the changed bit. This also means
that on average, a single bit change in the plaintext will effect
only a little more than half of the ciphertext.

I could _easily_ be wrong, but it seams reasonable to me that each bit
of plaintext would only have to diffuse to more than twice the number
of bits in the small block size for the large block cipher to be as
secure. It might even be less than the number of bits in the small
block size since the diffusion could be spread to all parts of the
larger block.

-wayne

--
Wayne Schlitt can not assert the truth of all statements in this
article and still be consistent.

========
Path:
news.io.com!imci4!newsfeed.internetmci.com!newsfeeder.gi.net!news.mid.net!mr.net!uunet!in2.uu.net!info.htcomp.net!NewsWatcher!user
From: wtshaw@htcomp.net (W T Shaw)
Newsgroups: sci.crypt
Subject: Re: Ciphers with *Huge* Block Size ?
Date: 26 Aug 1996 19:12:07 GMT
Organization: Another Netscape News Server User
Lines: 28
Message-ID: <wtshaw-2608961415530001@207.17.188.102>
References: <4vi4pb$fsq@elna.ethz.ch> <schneier-2408961653240001@news.winternet.com>
<WAYNE.96Aug24215404@backbone.uucp>
NNTP-Posting-Host: 207.17.188.102

In article <WAYNE.96Aug24215404@backbone.uucp>, wayne@cse.unl.edu wrote:
>
> I could _easily_ be wrong, but it seams reasonable to me that each bit
> of plaintext would only have to diffuse to more than twice the number
> of bits in the small block size for the large block cipher to be as
> secure. It might even be less than the number of bits in the small
> block size since the diffusion could be spread to all parts of the
> larger block.
>
Small blocks are compatible with small keys, unless some sort of chaining
takes place. Then the chaining of the later blocks might affect fewer and
fewer blocks, not a consistent effect at all. With larger blocks, a vast
amount of interaction between bits is readily available as a basic
characteristic of the algorithm. It is not necessary that all parts of the
block be affected by changing a single bit in the data.

Now looking at my pet algorithm, it really flies in the face of this
mixing wisdom, for while considerable diffusion of the key takes place,
the output does not at first glance seem to reflect any. The process does
have the desired affect, of course, of making the output exceedingly
difficult to correlate with the input while maintaining robustness of the
text, diminishing the effects of noise as a corrupting influence.
/\
wtshaw@htcomp.net   Mac Crypto Programs
 You should at least know how to use ROT13.
"Fhpprff vf n Wbhearl, Abg n Qrfgvangvba."
          http://www.htcomp.net/wts/
\/

========
Path:
news.io.com!arlut.utexas.edu!cs.utexas.edu!howland.erols.net!cam-news-hub1.bbnplanet.com!news.mathworks.com!fu-berlin.de!zrz.TU-Berlin.DE!news.dfn.de!news.gwdg.de!namu20.gwdg.de!lucks
From: Stefan Lucks <lucks@namu01.gwdg.de>
Newsgroups: sci.crypt
Subject: Re: Ciphers with *Huge* Block Size ?
Date: Sun, 25 Aug 1996 14:02:23 +0200
Organization: GWDG, Goettingen
Lines: 27

Message-ID: <Pine.OSF.3.91.960825135232.7490A-100000@namu20.gwdg.de>
References: <4vi4pb$fsq@elna.ethz.ch> <schneier-2408961653240001@news.winternet.com>
NNTP-Posting-Host: namu20.num.math.uni-goettingen.de
Mime-Version: 1.0
Content-Type: TEXT/PLAIN; charset=US-ASCII
X-Sender: lucks@namu20.gwdg.de
In-Reply-To: <schneier-2408961653240001@news.winternet.com>

On Sat, 24 Aug 1996, Bruce Schneier wrote:

> 2. It is a royal pain to analyze large block ciphers. It is hard enough
> to analyze constructions out of smaller primitives.

It is pretty easy to build block ciphers with huge blocks from
conventional cryptographic primitives, such as stream ciphers and hash
functions, and to analyze the block cipher's security.

You can even prove the block ciphers security (provided the building
blocks are secure).

Anderson and Biham did this with BEAR and LION. I,d propose BEAST, which
uses the same building blocks as BEAR and LION, but is faster. See the
"papers" section at my homepage for a paper describing BEAST, my
homepage's URL is "http://www.num.math.uni-goettingen.de/lucks/".

A link to Anderson's and Biham's paper has been given by someone else in
this thread.

Stefan Lucks Inst. f. NAM, Lotzestrasse 16-18, 37083 Goettingen, Germany
e-mail: lucks@math.uni-goettingen.de
home: http://www.num.math.uni-goettingen.de/lucks/
----- Wer einem Computer Unsinn erzaehlt, muss immer damit rechnen. -----

========
Path:
news.io.com!arlut.utexas.edu!cs.utexas.edu!chi-news.cic.net!newspump.sol.net!news.inc.net!uwm.edu!math.ohio-state.edu!howland.erols.net!newsfeed.internetmci.com!in3.uu.net!info.htcomp.net!NewsWatcher!user
From: wtshaw@htcomp.net (W T Shaw)
Newsgroups: sci.crypt
Subject: Re: Ciphers with *Huge* Block Size ?
Date: 26 Aug 1996 18:48:47 GMT
Organization: Another Netscape News Server User
Lines: 40
Message-ID: <wtshaw-2608961352320001@207.17.188.102>
References: <4vi4pb$fsq@elna.ethz.ch> <schneier-2408961653240001@news.winternet.com>
NNTP-Posting-Host: 207.17.188.102

In article <schneier-2408961653240001@news.winternet.com>,
schneier@counterpane.com (Bruce Schneier) wrote:

> There has not been a lot of research on LARGE block sizes, probably
> because they are so much harder to analyze.
>
> There are some serious problems with making a single block cipher with a
> block size in the multi-hundred byte range (as opposed to cascading a
> block cipher, stream cipher, and block cipher chaining in the other
> direction, which will do much the same thing).
>
> 1. It is very hard to get secure diffusion of every plaintext bit to
> every ciphertext bit. There are a lot more possibilities for good
> differentials to attack. On the other hand, you'll probably never get the
> volume of plaintext required for a differential attack.

Given a large block, the amount of diffusion becomes an option all its
own. Indeed, the keylength of the cipher can partly be scaled this way.
Contemporary key sizes would likely be at the bottom of the range.
>
> 2. It is a royal pain to analyze large block ciphers. It is hard enough
> to analyze constructions out of smaller primitives.
>
This sounds like a plus rather than a negative, but it depends on whether
you want it to be readily breakable or not, I prefer not.

> I kind of like the idea, but I worry about its security.

Like anything else new and/or different, large block ciphers need study to
see if there are basic flaws. Having best acquaintance of my own, I find
only a few people that few qualified, less after finding cryptoanalysis of
the pretty tough sledding.
>
Bill
/\
wtshaw@htcomp.net   Mac Crypto Programs
 You should at least know how to use ROT13.
"Fhpprff vf n Wbhearl, Abg n Qrfgvangvba."
http://www.htcomp.net/wts/
\/

========
Newsgroups: sci.crypt
Path:
news.io.com!arlut.utexas.edu!cs.utexas.edu!math.ohio-state.edu!uwm.edu!spool.mu.edu!news.sgi.com!wetware!news.dsndata.com!backbone!backbone!wayne

From: wayne@backbone.uucp (Wayne Schlitt)
Subject: Re: Ciphers with *Huge* Block Size ?
In-Reply-To: wtshaw@htcomp.net's message of 26 Aug 1996 18: 48:47 GMT
Message-ID: <WAYNE.96Aug26190836@backbone.uucp>
Sender: wayne@backbone (Wayne Schlitt)
Reply-To: wayne@cse.unl.edu
Organization: The Backbone Cabal
References: <4vi4pb$fsq@elna.ethz.ch> <schneier-2408961653240001@news.winternet.com>
<wtshaw-2608961352320001@207.17.188.102>
Date: Tue, 27 Aug 1996 01:08:36 GMT
Lines: 19

In article <wtshaw-2608961352320001@207.17.188.102> wtshaw@htcomp.net (W T Shaw)
writes:
> In article <schneier-2408961653240001@news.winternet.com>,
> schneier@counterpane.com (Bruce Schneier) wrote:
> > 2. It is a royal pain to analyze large block ciphers. It is hard enough
> > to analyze constructions out of smaller primitives.
> >
> This sounds like a plus rather than a negative, but it depends on whether
> you want it to be readily breakable or not, I prefer not.

Sounds like a _minus_ to me. You want to be able to _analyze_ a
cipher in order to _prove_ that it is strong.

-wayne

--
Wayne Schlitt can not assert the truth of all statements in this
article and still be consistent.

========
Path:
news.io.com!arlut.utexas.edu!cs.utexas.edu!howland.erols.net!cam-news-hub1.bbnplanet.com!news.mathworks.com!uunet!in3.uu.net!info.htcomp.net!NewsWatcher!user
From: wtshaw@htcomp.net (W T Shaw)
Newsgroups: sci.crypt
Subject: Re: Ciphers with *Huge* Block Size ?
Date: 27 Aug 1996 06:22:37 GMT
Organization: Another Netscape News Server User
Lines: 23
Message-ID: <wtshaw-2708960126130001@207.17.188.101>
References: <4vi4pb$fsq@elna.ethz.ch> <schneier-2408961653240001@news.winternet.com>
<wtshaw-2608961352320001@207.17.188.102> <WAYNE.96Aug26190836@backbone.uucp>
NNTP-Posting-Host: 207.17.188.101

In article <WAYNE.96Aug26190836@backbone.uucp>, wayne@cse.unl.edu wrote:

> In article <wtshaw-2608961352320001@207.17.188.102> wtshaw@htcomp.net (W
T Shaw) writes:
> > In article ,
> > schneier@counterpane.com (Bruce Schneier) wrote:
> > > 2. It is a royal pain to analyze large block ciphers. It is hard enough
> > > to analyze constructions out of smaller primitives.
> > >
> > This sounds like a plus rather than a negative, but it depends on whether
> > you want it to be readily breakable or not, I prefer not.

> Sounds like a _minus_ to me. You want to be able to _analyze_ a
> cipher in order to _prove_ that it is strong.
>
That is the idea, to make it hard to correlate the data. A strong cipher
should have this characteristic.
/\
wtshaw@htcomp.net   Mac Crypto Programs
 You should at least know how to use ROT13.
"Fhpprff vf n Wbhearl, Abg n Qrfgvangvba."
          http://www.htcomp.net/wts/
\/

========
Path:
news.io.com!news2.cais.net!news.cais.net!hunter.premier.net!news-res.gsl.net!news.gsl.net!swrinde!howland.erols.net!newsfeed.internetmci.com!in2.uu.net!ott.istar!istar.net!van.istar!west.istar!n1van.istar!van-bc!news.mindlink.net!uniserve!oronet!news.acsu.buffalo.edu!news.drenet.dnd.ca!crc-news.doc.ca!nott!cunews!usenet
From: Pat Morin <morin@scs.carleton.ca>
Newsgroups: sci.crypt
Subject: Re: Ciphers with *Huge* Block Size ?
Date: Sat, 24 Aug 1996 18:43:01 -0400
Organization: Carleton University
Lines: 37
Message-ID: <321F8575.4972@scs.carleton.ca>
References: <4vi4pb$fsq@elna.ethz.ch>
NNTP-Posting-Host: veda.scs.carleton.ca
Mime-Version: 1.0
Content-Type: text/plain; charset=us-ascii
Content-Transfer-Encoding: 7bit
X-Mailer: Mozilla 2.01 (X11; I; SunOS 5.5 sun4u)

Germano Caronni wrote:
>
> Hello everybody,
>
> current data encryption techniques usually encrypt a data stream or

```
> small blocks of data. While this makes perfect sense for communication
> where you need to encrypt and decrypt 'on the fly', it is perhaps not
> the optimal solution for archiving, and generally not the strongest way
> to secure data.  These are just idle thoughts, and I would be quite
> interested to hear what you think about it.
>
> Imagine a data file (or semantic entity, or whatever) being encrypted
> as a whole, which means that each of the output bits depends on each of
> the input bits (and naturally the secret key). This way, many kinds of
> attacks may be a bit less feasible e.g. imagine doing a known plaintext
> attack on something which simply can not be known in its entity, but of
> which certain parts are well known (e.g. contents of encapsulated IP
> headers, or headers of archived and encrypted mail) -- or imagine doing
> a linear CA attack when the relevant block size is about a Megabyte.
>
> Current block cipher chaining methods (CBC, OFB, CFB, ...) do not offer
> this property, the best result is that the last encrypted 'block'
> depends on all previous elements.
>
> Comments to this one?

Look at the BEAR and LION ciphers developed by Ross Anderson and
Eli Biham.  They're fast, easy to implement, and have some nice
security properties.

ftp://ftp.cl.cam.ac.uk/users/rja14/bear-lion.ps.Z

If you're really interested, you can also look at AARDVARK.

http://www.scs.carleton.ca/~morin/sac96/AARDVARK/AARDVARK.ps

Pat
```

Proceedings of the 2 nd Conference on Applied Statistics in Agriculture
Kansas State University, Dwight D. Eisenhower Hall, Manhattan, Kansas, April 26 - 28, 1992
Editors: Gerald A. Sullivan, Ronald H. Trewin, Clifford A. Bond, Ronald D. Lacewell, Ronald D. Seifert

Estimating Population from Repetitions in Accumulated Random Samples

Terry Ritter
Cryptech, Inc.
3925 West Braker Lane, Austin, Texas 78759, (512) 343-0484

ABSTRACT: It is desired to estimate the number of unique values in a data population by collecting random samples generated at that same data population. Such data might be cryptographic message keys, sale dates, or even the ZIP codes of customers. Although the simple Poisson model for repetitions in such samples is well known, it is not generally known that the Poisson model is quite robust in the face of non-random data. Although the Poisson model is also well known for its "birthday paradox" property, this paper shows that the Poisson model is also robust in the face of non-random data in this application. Although the Poisson model is robust, it is not perfect, and this paper also shows how to compensate for its imperfections. Although the Poisson model is well known, it is not generally known that the Poisson model can be used to estimate the number of unique values in a data population, and this paper reviews the relevant literature. These previous findings suggest that the Poisson model is a very reasonable choice for estimating population from repetitions in accumulated random samples, and this paper reviews the relevant literature. These previous findings suggest that the Poisson model is a very reasonable choice for estimating population from repetitions in accumulated random samples, and this paper reviews the relevant literature.

INTRODUCTION

Cryptographic systems often need "random" numbers for various applications. Unfortunately, computers are not very good at generating random numbers. Some random numbers are valuable because they allow a cryptanalyst to break the system. Such valuable random numbers are the exception to the general rule, which generally needed "random" numbers. Although, Normally, such almost random numbers are adequate for most cryptographic applications. These random numbers are the exception to the general rule, which generally needed "random" numbers. Although, Normally, such almost random numbers are adequate for most cryptographic applications. These random numbers are the exception to the general rule, which generally needed "random" numbers. Although, Normally, such almost random numbers are adequate for most cryptographic applications.

Random numbers are valuable because they allow a cryptanalyst to break the system. Such valuable random numbers are the exception to the general rule, which generally needed "random" numbers. Although, Normally, such almost random numbers are adequate for most cryptographic applications. These random numbers are the exception to the general rule, which generally needed "random" numbers. Although, Normally, such almost random numbers are adequate for most cryptographic applications.

Random numbers are valuable because they allow a cryptanalyst to break the system. Such valuable random numbers are the exception to the general rule, which generally needed "random" numbers. Although, Normally, such almost random numbers are adequate for most cryptographic applications. These random numbers are the exception to the general rule, which generally needed "random" numbers. Although, Normally, such almost random numbers are adequate for most cryptographic applications.

THE BIRTHDAY PARADOX

The birthday paradox is a well known phenomenon in which there is at least one of the relative frequency of the birthday paradox. For example, Feller (1968), Feller (1968), Feller (1968).

Let the first question we have proposed. Then, the birthday paradox is a well known phenomenon in which there is at least one of the relative frequency of the birthday paradox. For example, Feller (1968), Feller (1968), Feller (1968).

Let us consider the birthday paradox. Let us consider the birthday paradox. Let us consider the birthday paradox. Let us consider the birthday paradox. Let us consider the birthday paradox.

Let us consider the birthday paradox. Let us consider the birthday paradox. Let us consider the birthday paradox. Let us consider the birthday paradox. Let us consider the birthday paradox.

Let us consider the birthday paradox. Let us consider the birthday paradox. Let us consider the birthday paradox. Let us consider the birthday paradox. Let us consider the birthday paradox.

Fig. 1 Birthday Probability Logic. For n=1 and, the first values. So, then we accumulate, if n=2 and, the second value, then the probability of a match is 1/365. If n=3 and, the third value, then the probability of a match is 3/365. If n=4 and, the fourth value, then the probability of a match is 6/365. If n=5 and, the fifth value, then the probability of a match is 10/365. If n=6 and, the sixth value, then the probability of a match is 15/365. If n=7 and, the seventh value, then the probability of a match is 21/365. If n=8 and, the eighth value, then the probability of a match is 28/365. If n=9 and, the ninth value, then the probability of a match is 36/365. If n=10 and, the tenth value, then the probability of a match is 45/365. If n=11 and, the eleventh value, then the probability of a match is 55/365. If n=12 and, the twelfth value, then the probability of a match is 66/365. If n=13 and, the thirteenth value, then the probability of a match is 78/365. If n=14 and, the fourteenth value, then the probability of a match is 91/365. If n=15 and, the fifteenth value, then the probability of a match is 105/365. If n=16 and, the sixteenth value, then the probability of a match is 120/365. If n=17 and, the seventh value, then the probability of a match is 136/365. If n=18 and, the eighth value, then the probability of a match is 153/365. If n=19 and, the ninth value, then the probability of a match is 171/365. If n=20 and, the tenth value, then the probability of a match is 190/365. If n=21 and, the eleventh value, then the probability of a match is 210/365. If n=22 and, the twelfth value, then the probability of a match is 231/365. If n=23 and, the thirteenth value, then the probability of a match is 253/365. If n=24 and, the fourteenth value, then the probability of a match is 276/365. If n=25 and, the fifteenth value, then the probability of a match is 300/365. If n=26 and, the sixteenth value, then the probability of a match is 325/365. If n=27 and, the seventh value, then the probability of a match is 351/365. If n=28 and, the eighth value, then the probability of a match is 378/365. If n=29 and, the ninth value, then the probability of a match is 406/365. If n=30 and, the tenth value, then the probability of a match is 435/365. If n=31 and, the eleventh value, then the probability of a match is 465/365. If n=32 and, the twelfth value, then the probability of a match is 496/365. If n=33 and, the thirteenth value, then the probability of a match is 528/365. If n=34 and, the fourteenth value, then the probability of a match is 561/365. If n=35 and, the fifteenth value, then the probability of a match is 595/365. If n=36 and, the sixteenth value, then the probability of a match is 630/365. If n=37 and, the seventh value, then the probability of a match is 666/365. If n=38 and, the eighth value, then the probability of a match is 703/365. If n=39 and, the ninth value, then the probability of a match is 741/365. If n=40 and, the tenth value, then the probability of a match is 780/365. If n=41 and, the eleventh value, then the probability of a match is 820/365. If n=42 and, the twelfth value, then the probability of a match is 861/365. If n=43 and, the thirteenth value, then the probability of a match is 903/365. If n=44 and, the fourteenth value, then the probability of a match is 946/365. If n=45 and, the fifteenth value, then the probability of a match is 990/365. If n=46 and, the sixteenth value, then the probability of a match is 1035/365. If n=47 and, the seventh value, then the probability of a match is 1081/365. If n=48 and, the eighth value, then the probability of a match is 1128/365. If n=49 and, the ninth value, then the probability of a match is 1176/365. If n=50 and, the tenth value, then the probability of a match is 1225/365. If n=51 and, the eleventh value, then the probability of a match is 1275/365. If n=52 and, the twelfth value, then the probability of a match is 1326/365. If n=53 and, the thirteenth value, then the probability of a match is 1378/365. If n=54 and, the fourteenth value, then the probability of a match is 1431/365. If n=55 and, the fifteenth value, then the probability of a match is 1485/365. If n=56 and, the sixteenth value, then the probability of a match is 1540/365. If n=57 and, the seventh value, then the probability of a match is 1596/365. If n=58 and, the eighth value, then the probability of a match is 1653/365. If n=59 and, the ninth value, then the probability of a match is 1711/365. If n=60 and, the tenth value, then the probability of a match is 1770/365. If n=61 and, the eleventh value, then the probability of a match is 1830/365. If n=62 and, the twelfth value, then the probability of a match is 1891/365. If n=63 and, the thirteenth value, then the probability of a match is 1953/365. If n=64 and, the fourteenth value, then the probability of a match is 2016/365. If n=65 and, the fifteenth value, then the probability of a match is 2080/365. If n=66 and, the sixteenth value, then the probability of a match is 2145/365. If n=67 and, the seventh value, then the probability of a match is 2211/365. If n=68 and, the eighth value, then the probability of a match is 2278/365. If n=69 and, the ninth value, then the probability of a match is 2346/365. If n=70 and, the tenth value, then the probability of a match is 2415/365. If n=71 and, the eleventh value, then the probability of a match is 2485/365. If n=72 and, the twelfth value, then the probability of a match is 2556/365. If n=73 and, the thirteenth value, then the probability of a match is 2628/365. If n=74 and, the fourteenth value, then the probability of a match is 2701/365. If n=75 and, the fifteenth value, then the probability of a match is 2775/365. If n=76 and, the sixteenth value, then the probability of a match is 2850/365. If n=77 and, the seventh value, then the probability of a match is 2926/365. If n=78 and, the eighth value, then the probability of a match is 3003/365. If n=79 and, the ninth value, then the probability of a match is 3081/365. If n=80 and, the tenth value, then the probability of a match is 3160/365. If n=81 and, the eleventh value, then the probability of a match is 3240/365. If n=82 and, the twelfth value, then the probability of a match is 3321/365. If n=83 and, the thirteenth value, then the probability of a match is 3403/365. If n=84 and, the fourteenth value, then the probability of a match is 3486/365. If n=85 and, the fifteenth value, then the probability of a match is 3570/365. If n=86 and, the sixteenth value, then the probability of a match is 3655/365. If n=87 and, the seventh value, then the probability of a match is 3741/365. If n=88 and, the eighth value, then the probability of a match is 3828/365. If n=89 and, the ninth value, then the probability of a match is 3916/365. If n=90 and, the tenth value, then the probability of a match is 4005/365. If n=91 and, the eleventh value, then the probability of a match is 4095/365. If n=92 and, the twelfth value, then the probability of a match is 4186/365. If n=93 and, the thirteenth value, then the probability of a match is 4278/365. If n=94 and, the fourteenth value, then the probability of a match is 4371/365. If n=95 and, the fifteenth value, then the probability of a match is 4465/365. If n=96 and, the sixteenth value, then the probability of a match is 4560/365. If n=97 and, the seventh value, then the probability of a match is 4656/365. If n=98 and, the eighth value, then the probability of a match is 4753/365. If n=99 and, the ninth value, then the probability of a match is 4851/365. If n=100 and, the tenth value, then the probability of a match is 4950/365. If n=101 and, the eleventh value, then the probability of a match is 5050/365. If n=102 and, the twelfth value, then the probability of a match is 5151/365. If n=103 and, the thirteenth value, then the probability of a match is 5253/365. If n=104 and, the fourteenth value, then the probability of a match is 5356/365. If n=105 and, the fifteenth value, then the probability of a match is 5460/365. If n=106 and, the sixteenth value, then the probability of a match is 5565/365. If n=107 and, the seventh value, then the probability of a match is 5671/365. If n=108 and, the eighth value, then the probability of a match is 5778/365. If n=109 and, the ninth value, then the probability of a match is 5886/365. If n=110 and, the tenth value, then the probability of a match is 5995/365. If n=111 and, the eleventh value, then the probability of a match is 6105/365. If n=112 and, the twelfth value, then the probability of a match is 6216/365. If n=113 and, the thirteenth value, then the probability of a match is 6328/365. If n=114 and, the fourteenth value, then the probability of a match is 6441/365. If n=115 and, the fifteenth value, then the probability of a match is 6555/365. If n=116 and, the sixteenth value, then the probability of a match is 6670/365. If n=117 and, the seventh value, then the probability of a match is 6786/365. If n=118 and, the eighth value, then the probability of a match is 6903/365. If n=119 and, the ninth value, then the probability of a match is 7021/365. If n=120 and, the tenth value, then the probability of a match is 7140/365. If n=121 and, the eleventh value, then the probability of a match is 7260/365. If n=122 and, the twelfth value, then the probability of a match is 7381/365. If n=123 and, the thirteenth value, then the probability of a match is 7503/365. If n=124 and, the fourteenth value, then the probability of a match is 7626/365. If n=125 and, the fifteenth value, then the probability of a match is 7750/365. If n=126 and, the sixteenth value, then the probability of a match is 7875/365. If n=127 and, the seventh value, then the probability of a match is 8001/365. If n=128 and, the eighth value, then the probability of a match is 8128/365. If n=129 and, the ninth value, then the probability of a match is 8256/365. If n=130 and, the tenth value, then the probability of a match is 8385/365. If n=131 and, the eleventh value, then the probability of a match is 8515/365. If n=132 and, the twelfth value, then the probability of a match is 8646/365. If n=133 and, the thirteenth value, then the probability of a match is 8778/365. If n=134 and, the fourteenth value, then the probability of a match is 8911/365. If n=135 and, the fifteenth value, then the probability of a match is 9045/365. If n=136 and, the sixteenth value, then the probability of a match is 9180/365. If n=137 and, the seventh value, then the probability of a match is 9316/365. If n=138 and, the eighth value, then the probability of a match is 9453/365. If n=139 and, the ninth value, then the probability of a match is 9591/365. If n=140 and, the tenth value, then the probability of a match is 9730/365. If n=141 and, the eleventh value, then the probability of a match is 9870/365. If n=142 and, the twelfth value, then the probability of a match is 10011/365. If n=143 and, the thirteenth value, then the probability of a match is 10153/365. If n=144 and, the fourteenth value, then the probability of a match is 10296/365. If n=145 and, the fifteenth value, then the probability of a match is 10440/365. If n=146 and, the sixteenth value, then the probability of a match is 10585/365. If n=147 and, the seventh value, then the probability of a match is 10731/365. If n=148 and, the eighth value, then the probability of a match is 10878/365. If n=149 and, the ninth value, then the probability of a match is 11026/365. If n=150 and, the tenth value, then the probability of a match is 11175/365. If n=151 and, the eleventh value, then the probability of a match is 11325/365. If n=152 and, the twelfth value, then the probability of a match is 11476/365. If n=153 and, the thirteenth value, then the probability of a match is 11628/365. If n=154 and, the fourteenth value, then the probability of a match is 11781/365. If n=155 and, the fifteenth value, then the probability of a match is 11935/365. If n=156 and, the sixteenth value, then the probability of a match is 12090/365. If n=157 and, the seventh value, then the probability of a match is 12246/365. If n=158 and, the eighth value, then the probability of a match is 12403/365. If n=159 and, the ninth value, then the probability of a match is 12561/365. If n=160 and, the tenth value, then the probability of a match is 12720/365. If n=161 and, the eleventh value, then the probability of a match is 12880/365. If n=162 and, the twelfth value, then the probability of a match is 13041/365. If n=163 and, the thirteenth value, then the probability of a match is 13203/365. If n=164 and, the fourteenth value, then the probability of a match is 13366/365. If n=165 and, the fifteenth value, then the probability of a match is 13530/365. If n=166 and, the sixteenth value, then the probability of a match is 13695/365. If n=167 and, the seventh value, then the probability of a match is 13861/365. If n=168 and, the eighth value, then the probability of a match is 14028/365. If n=169 and, the ninth value, then the probability of a match is 14196/365. If n=170 and, the tenth value, then the probability of a match is 14365/365. If n=171 and, the eleventh value, then the probability of a match is 14535/365. If n=172 and, the twelfth value, then the probability of a match is 14706/365. If n=173 and, the thirteenth value, then the probability of a match is 14878/365. If n=174 and, the fourteenth value, then the probability of a match is 15051/365. If n=175 and, the fifteenth value, then the probability of a match is 15225/365. If n=176 and, the sixteenth value, then the probability of a match is 15400/365. If n=177 and, the seventh value, then the probability of a match is 15576/365. If n=178 and, the eighth value, then the probability of a match is 15753/365. If n=179 and, the ninth value, then the probability of a match is 15931/365. If n=180 and, the tenth value, then the probability of a match is 16110/365. If n=181 and, the eleventh value, then the probability of a match is 16290/365. If n=182 and, the twelfth value, then the probability of a match is 16471/365. If n=183 and, the thirteenth value, then the probability of a match is 16653/365. If n=184 and, the fourteenth value, then the probability of a match is 16836/365. If n=185 and, the fifteenth value, then the probability of a match is 17020/365. If n=186 and, the sixteenth value, then the probability of a match is 17205/365. If n=187 and, the seventh value, then the probability of a match is 17391/365. If n=188 and, the eighth value, then the probability of a match is 17578/365. If n=189 and, the ninth value, then the probability of a match is 17766/365. If n=190 and, the tenth value, then the probability of a match is 17955/365. If n=191 and, the eleventh value, then the probability of a match is 18145/365. If n=192 and, the twelfth value, then the probability of a match is 18336/365. If n=193 and, the thirteenth value, then the probability of a match is 18528/365. If n=194 and, the fourteenth value, then the probability of a match is 18721/365. If n=195 and, the fifteenth value, then the probability of a match is 18915/365. If n=196 and, the sixteenth value, then the probability of a match is 19110/365. If n=197 and, the seventh value, then the probability of a match is 19306/365. If n=198 and, the eighth value, then the probability of a match is 19503/365. If n=199 and, the ninth value, then the probability of a match is 19701/365. If n=200 and, the tenth value, then the probability of a match is 19900/365. If n=201 and, the eleventh value, then the probability of a match is 20100/365. If n=202 and, the twelfth value, then the probability of a match is 20301/365. If n=203 and, the thirteenth value, then the probability of a match is 20503/365. If n=204 and, the fourteenth value, then the probability of a match is 20706/365. If n=205 and, the fifteenth value, then the probability of a match is 20910/365. If n=206 and, the sixteenth value, then the probability of a match is 21115/365. If n=207 and, the seventh value, then the probability of a match is 21321/365. If n=208 and, the eighth value, then the probability of a match is 21528/365. If n=209 and, the ninth value, then the probability of a match is 21736/365. If n=210 and, the tenth value, then the probability of a match is 21945/365. If n=211 and, the eleventh value, then the probability of a match is 22155/365. If n=212 and, the twelfth value, then the probability of a match is 22366/365. If n=213 and, the thirteenth value, then the probability of a match is 22578/365. If n=214 and, the fourteenth value, then the probability of a match is 22791/365. If n=215 and, the fifteenth value, then the probability of a match is 23005/365. If n=216 and, the sixteenth value, then the probability of a match is 23220/365. If n=217 and, the seventh value, then the probability of a match is 23436/365. If n=218 and, the eighth value, then the probability of a match is 23653/365. If n=219 and, the ninth value, then the probability of a match is 23871/365. If n=220 and, the tenth value, then the probability of a match is 24090/365. If n=221 and, the eleventh value, then the probability of a match is 24310/365. If n=222 and, the twelfth value, then the probability of a match is 24531/365. If n=223 and, the thirteenth value, then the probability of a match is 24753/365. If n=224 and, the fourteenth value, then the probability of a match is 24976/365. If n=225 and, the fifteenth value, then the probability of a match is 25200/365. If n=226 and, the sixteenth value, then the probability of a match is 25425/365. If n=227 and, the seventh value, then the probability of a match is 25651/365. If n=228 and, the eighth value, then the probability of a match is 25878/365. If n=229 and, the ninth value, then the probability of a match is 26106/365. If n=230 and, the tenth value, then the probability of a match is 26335/365. If n=231 and, the eleventh value, then the probability of a match is 26565/365. If n=232 and, the twelfth value, then the probability of a match is 26796/365. If n=233 and, the thirteenth value, then the probability of a match is 27028/365. If n=234 and, the fourteenth value, then the probability of a match is 27261/365. If n=235 and, the fifteenth value, then the probability of a match is 27495/365. If n=236 and, the sixteenth value, then the probability of a match is 27730/365. If n=237 and, the seventh value, then the probability of a match is 27966/365. If n=238 and, the eighth value, then the probability of a match is 28203/365. If n=239 and, the ninth value, then the probability of a match is 28441/365. If n=240 and, the tenth value, then the probability of a match is 28680/365. If n=241 and, the eleventh value, then the probability of a match is 28920/365. If n=242 and, the twelfth value, then the probability of a match is 29161/365. If n=243 and, the thirteenth value, then the probability of a match is 29403/365. If n=244 and, the fourteenth value, then the probability of a match is 29646/365. If n=245 and, the fifteenth value, then the probability of a match is 29890/365. If n=246 and, the sixteenth value, then the probability of a match is 30135/365. If n=247 and, the seventh value, then the probability of a match is 30381/365. If n=248 and, the eighth value, then the probability of a match is 30628/365. If n=249 and, the ninth value, then the probability of a match is 30876/365. If n=250 and, the tenth value, then the probability of a match is 31125/365. If n=251 and, the eleventh value, then the probability of a match is 31375/365. If n=252 and, the twelfth value, then the probability of a match is 31626/365. If n=253 and, the thirteenth value, then the probability of a match is 31878/365. If n=254 and, the fourteenth value, then the probability of a match is 32131/365. If n=255 and, the fifteenth value, then the probability of a match is 32385/365. If n=256 and, the sixteenth value, then the probability of a match is 32640/365. If n=257 and, the seventh value, then the probability of a match is 32896/365. If n=258 and, the eighth value, then the probability of a match is 33153/365. If n=259 and, the ninth value, then the probability of a match is 33411/365. If n=260 and, the tenth value, then the probability of a match is 33670/365. If n=261 and, the eleventh value, then the probability of a match is 33930/365. If n=262 and, the twelfth value, then the probability of a match is 34191/365. If n=263 and, the thirteenth value, then the probability of a match is 34453/365. If n=264 and, the fourteenth value, then the probability of a match is 34716/365. If n=265 and, the fifteenth value, then the probability of a match is 34980/365. If n=266 and, the sixteenth value, then the probability of a match is 35245/365. If n=267 and, the seventh value, then the probability of a match is 35511/365. If n=268 and, the eighth value, then the probability of a match is 35778/365. If n=269 and, the ninth value, then the probability of a match is 36046/365. If n=270 and, the tenth value, then the probability of a match is 36315/365. If n=271 and, the eleventh value, then the probability of a match is 36585/365. If n=272 and, the twelfth value, then the probability of a match is 36856/365. If n=273 and, the thirteenth value, then the probability of a match is 37128/365. If n=274 and, the fourteenth value, then the probability of a match is 37401/365. If n=275 and, the fifteenth value, then the probability of a match is 37675/365. If n=276 and, the sixteenth value, then the probability of a match is 37950/365. If n=277 and, the seventh value, then the probability of a match is 38226/365. If n=278 and, the eighth value, then the probability of a match is 38503/365. If n=279 and, the ninth value, then the probability of a match is 38781/365. If n=280 and, the tenth value, then the probability of a match is 39060/365. If n=281 and, the eleventh value, then the probability of a match is 39340/365. If n=282 and, the twelfth value, then the probability of a match is 39621/365. If n=283 and, the thirteenth value, then the probability of a match is 39903/365. If n=284 and, the fourteenth value, then the probability of a match is 40186/365. If n=285 and, the fifteenth value, then the probability of a match is 40470/365. If n=286 and, the sixteenth value, then the probability of a match is 40755/365. If n=287 and, the seventh value, then the probability of a match is 41041/365. If n=288 and, the eighth value, then the probability of a match is 41328/365. If n=289 and, the ninth value, then the probability of a match is 41616/365. If n=290 and, the tenth value, then the probability of a match is 41905/365. If n=291 and, the eleventh value, then the probability of a match is 42195/365. If n=292 and, the twelfth value, then the probability of a match is 42486/365. If n=293 and, the thirteenth value, then the probability of a match is 42778/365. If n=294 and, the fourteenth value, then the probability of a match is 43071/365. If n=295 and, the fifteenth value, then the probability of a match is 43365/365. If n=296 and, the sixteenth value, then the probability of a match is 43660/365. If n=297 and, the seventh value, then the probability of a match is 43956/365. If n=298 and, the eighth value, then the probability of a match is 44253/365. If n=299 and, the ninth value, then the probability of a match is 44551/365. If n=300 and, the tenth value, then the probability of a match is 44850/365. If n=301 and, the eleventh value, then the probability of a match is 45150/365. If n=302 and, the twelfth value, then the probability of a match is 45451/365. If n=303 and, the thirteenth value, then the probability of a match is 45753/365. If n=304 and, the fourteenth value, then the probability of a match is 46056/365. If n=305 and, the fifteenth value, then the probability of a match is 46360/365. If n=306 and, the sixteenth value, then the probability of a match is 46665/365. If n=307 and, the seventh value, then the probability of a match is 46971/365. If n=308 and, the eighth value, then the probability of a match is 47278/365. If n=309 and, the ninth value, then the probability of a match is 47586/365. If n=310 and, the tenth value, then the probability of a match is 47895/365. If n=311 and, the eleventh value, then the probability of a match is 48205/365. If n=312 and, the twelfth value, then the probability of a match is 48516/365. If n=313 and, the thirteenth value, then the probability of a match is 48828/365. If n=314 and, the fourteenth value, then the probability of a match is 49141/365. If n=315 and, the fifteenth value, then the probability of a match is 49455/365. If n=316 and, the sixteenth value, then the probability of a match is 49770/365. If n=317 and, the seventh value, then the probability of a match is 50086/365. If n=318 and, the eighth value, then the probability of a match is 50403/365. If n=319 and, the ninth value, then the probability of a match is 50721/365. If n=320 and, the tenth value, then the probability of a match is 51040/365. If n=321 and, the eleventh value, then the probability of a match is 51360/365. If n=322 and, the twelfth value, then the probability of a match is 51681/365. If n=323 and, the thirteenth value, then the probability of a match is 52003/365. If n=324 and, the fourteenth value, then the probability of a match is 52326/365. If n=325 and, the fifteenth value, then the probability of a match is 52650/365. If n=326 and, the sixteenth value, then the probability of a match is 52975/365. If n=327 and, the seventh value, then the probability of a match is 53301/365. If n=328 and, the eighth value, then the probability of a match is 53628/365. If n=329 and, the ninth value, then the probability of a match is 53956/365. If n=330 and, the tenth value, then the probability of a match is 54285/365. If n=331 and, the eleventh value, then the probability of a match is 54615/365. If n=332 and, the twelfth value, then the probability of a match is 54946/365. If n=333 and, the thirteenth value, then the probability of a match is 55278/365. If n=334 and, the fourteenth value, then the probability of a match is 55611/365. If n=335 and, the fifteenth value, then the probability of a match is 55945/365. If n=336 and, the sixteenth value, then the probability of a match is 56280/365. If n=337 and, the seventh value, then the probability of a match is 56616/365. If n=338 and, the eighth value, then the probability of a match is 56953/365. If n=339 and, the ninth value, then the probability of a match is 57291/365. If n=340 and, the tenth value, then the probability of a match is 57630/365. If n=341 and, the eleventh value, then the probability of a match is 57970/365. If n=342 and, the twelfth value, then the probability of a match is 58311/365. If n=343 and, the thirteenth value, then the probability of a match is 58653/365. If n=344 and, the fourteenth value, then the probability of a match is 58996/365. If n=345 and, the fifteenth value, then the probability of a match is 59340/365. If n=346 and, the sixteenth value, then the probability of a match is 59685/365. If n=347 and, the seventh value, then the probability of a match is 60031/365. If n=348 and, the eighth value, then the probability of a match is 60378/365. If n=349 and, the ninth value, then the probability of a match is 60726/365. If n=350 and, the tenth value, then the probability of a match is 61075/365. If n=351 and, the eleventh value, then the probability of a match is 61425/365. If n=352 and, the twelfth value, then the probability of a match is 61776/365. If n=353 and, the thirteenth value, then the probability of a match is 62128/365. If n=354 and, the fourteenth value, then the probability of a match is 62481/365. If n=355 and, the fifteenth value, then the probability of a match is 62835/365. If n=356 and, the sixteenth value, then the probability of a match is 63190/365. If n=357 and, the seventh value, then the probability of a match is 63546/365. If n=358 and, the eighth value, then the probability of a match is 63903/365. If n=359 and, the ninth value, then the probability of a match is 64261/365. If n=360 and, the tenth value, then the probability of a match is 64620/365. If n=361 and, the eleventh value, then the probability of a match is 64980/365. If n=362 and, the twelfth value, then the probability of a match is 65341/365. If n=363 and, the thirteenth value, then the probability of a match is 65703/365. If n=364 and, the fourteenth value, then the probability of a match is 66066/365. If n=365 and, the fifteenth value, then the probability of a match is 66430/365. If n=366 and, the sixteenth value, then the probability of a match is 66795/365. If n=367 and, the seventh value, then the probability of a match is 67161/365. If n=368 and, the eighth value, then the probability of a match is 67528/365. If n=369 and, the ninth value, then the probability of a match is 67896/365. If n=370 and, the tenth value, then the probability of a match is 68265/365. If n=371 and, the eleventh value, then the probability of a match is 68635/365. If n=372 and, the twelfth value, then the probability of a match is 69006/365. If n=373 and, the thirteenth value, then the probability of a match is 69378/365. If n=374 and, the fourteenth value, then the probability of a match is 69751/365. If n=375 and, the fifteenth value, then the probability of a match is 70125/365. If n=376 and, the sixteenth value, then the probability of a match is 70500/365. If n=377 and, the seventh value, then the probability of a match is 70876/365. If n=378 and, the eighth value, then the probability of a match is 71253/365. If n=379 and, the ninth value, then the probability of a match is 71631/365. If n=380 and, the tenth value, then the probability of a match is 72010/365. If n=381 and, the eleventh value, then the probability of a match is 72390/365. If n=382 and, the twelfth value, then the probability of a match is 72771/365. If n=383 and, the thirteenth value, then the probability of a match is 73153/365. If n=384 and, the fourteenth value, then the probability of a match is 73536/365. If n=385 and, the fifteenth value, then the probability of a match is 73920/365. If n=386 and, the sixteenth value, then the probability of a match is 74305/365. If n=387 and, the seventh value, then the probability of a match is 74691/365. If n=388 and, the eighth value, then the probability of a match is 75078/365. If n=389 and, the ninth value, then the probability of a match is 75466/365. If n=390 and, the tenth value, then the probability of a match is 75855/365. If n=391 and, the eleventh value, then the probability of a match is 76245/365. If n=392 and, the twelfth value, then the probability of a match is 76636/365. If n=393 and, the thirteenth value, then the probability of a match is 77028/365. If n=394 and, the fourteenth value, then the probability of a match is 77421/365. If n=395 and, the fifteenth value, then the probability of a match is 77815/365. If n=396 and, the sixteenth value, then the probability of a match is 78210/365. If n=397 and, the seventh value, then the probability of a match is 78606/365. If n=398 and, the eighth value, then the probability of a match is 79003/365. If n=399 and, the ninth value, then the probability of a match is 79401/365. If n=400 and, the tenth value, then the probability of a match is 79800/365. If n=401 and, the eleventh value, then the probability of a match is 80200/365. If n=402 and, the twelfth value, then the probability of a match is 80601/365. If n=403 and, the thirteenth value, then the probability of a match is 81003/365. If n=404 and, the fourteenth value, then the probability of a match is 81406/365. If n=405 and, the fifteenth value, then the probability of a match is 81810/365. If n=406 and, the sixteenth value, then the probability of a match is 82215/365. If n=407 and, the seventh value, then the probability of a match is 82621/365. If n=408 and, the eighth value, then the probability of a match is 83028/365. If n=409 and, the ninth value, then the probability of a match is 83436/365. If n=410 and, the tenth value, then the probability of a match is 83845/365. If n=411 and, the eleventh value, then the probability of a match is 84255/365. If n=412 and, the twelfth value, then the probability of a match is 84666/365. If n=413 and, the thirteenth value, then the probability of a match is 85078/365. If n=414 and, the fourteenth value, then the probability of a match is 85491/365. If n=415 and, the fifteenth value, then the probability of a match is 85905/365. If n=416 and, the sixteenth value, then the probability of a match is 86320/365. If n=417 and, the seventh value, then the probability of a match is 86736/365. If n=418 and, the eighth value, then the probability of a match is 87153/365. If n=419 and, the ninth value, then the probability of a match is 87571/365. If n=420 and, the tenth value, then the probability of a match is 87990/365. If n=421 and, the eleventh value, then the probability of a match is 88410/365. If n=422 and, the twelfth value, then the probability of a match is 88831/365. If n=423 and, the thirteenth value, then the probability of a match is 89253/365. If n=424 and, the fourteenth value, then the probability of a match is 89676/365. If n=425 and, the fifteenth value, then the probability of a match is 90100/365. If n=426 and, the sixteenth value, then the probability of a match is 90525/365. If n=427 and, the seventh value, then the probability of a match is 90951/365. If n=428 and, the eighth value, then the probability of a match is 91378/365. If n=429 and, the ninth value, then the probability of a match is 91806/365. If n=430 and, the tenth value, then the probability of a match is 92235/365. If n=431 and, the eleventh value, then the probability of a match is 92665/365. If n=432 and, the twelfth value, then the probability of a match is 93096/365. If n=433 and, the thirteenth value, then the probability of a match is 93528/365. If n=434 and, the fourteenth value, then the probability of a match is 93961/365. If n=435 and, the fifteenth value, then the probability of a match is 94395/365. If n=436 and, the sixteenth value, then the probability of a match is 94830/365. If n=437 and, the seventh value, then the probability of a match is 95266/365. If n=438 and, the eighth value, then the probability of a match is 95703/365. If n=439 and, the ninth value, then the probability of a match is 96141/365. If n=440 and, the tenth value, then the probability of a match is 96580/365. If n=441 and, the eleventh value, then the probability of a match is 97020/365. If n=442 and, the twelfth value, then the probability of a match is 97461/365. If n=443 and, the thirteenth value, then the probability of a match is 97903/365. If n=444 and, the fourteenth value, then the probability of a match is 98346/365. If n=445 and, the fifteenth value, then the probability of a match is 98790/365. If n=446 and, the sixteenth value, then the probability of a match is 99235/365. If n=447 and, the seventh value, then the probability of a match is 99681/365. If n=448 and, the eighth value, then the probability of a match is 100128/365. If n=449 and, the ninth value, then the probability of a match is 100576/365. If n=450 and, the tenth value, then the probability of a match is 101025/365. If n=451 and, the eleventh value, then the probability of a match is 101475/365. If n=452 and, the twelfth value, then the probability of a match is 101926/365. If n=453 and, the thirteenth value, then the probability of a match is 102378/365. If n=454 and, the fourteenth value, then the probability of a match is 102831/365. If n=455 and, the fifteenth value, then the probability of a match is 103285/365. If n=456 and, the sixteenth value, then the probability of a match is 103740/365. If n=457 and, the seventh value, then the probability of a match is 104196/365. If n=458 and, the eighth value, then the probability of a match is 104653/365. If n=459 and, the ninth value, then the probability of a match is 105111/365. If n=460 and, the tenth value, then the probability of a match is 105570/365. If n=461 and, the eleventh value, then the probability of a match is 106030/365. If n=462 and, the twelfth value, then the probability of a match is 106491/365. If n=463 and, the thirteenth value, then the probability of a match is 106953/365. If n=464 and, the fourteenth value, then the probability of a match is 107416/365. If n=465 and, the fifteenth value, then the probability of a match is 107880/365. If n=466 and, the sixteenth value, then the probability of a match is 108345/365. If n=467 and, the seventh value, then the probability of a match is 108811/365. If n=468 and, the eighth value, then the probability of a match is 109278/365. If n=469 and, the ninth value, then the probability of a match is 109746/365. If n=470 and, the tenth value, then the probability of a match is 110215/365. If n=471 and, the eleventh value, then the probability of a match is 110685/365. If n=472 and, the twelfth value, then the probability of a match is 111156/365. If n=473 and, the thirteenth value, then the probability of a match is 111628/365. If n=474 and, the fourteenth value, then the probability of a match is 112101/365. If n=475 and, the fifteenth value, then the probability of a match is 112575/365. If n=476 and, the sixteenth value, then the probability of a match is 113050/365. If n=477 and, the seventh value, then the probability of a match is 113526/365. If n=478 and, the eighth value, then the probability of a match is 114003/365. If n=479 and, the ninth value, then the probability of a match is 114481/365. If n=480 and, the tenth value, then the probability of a match is 114960/365. If n=481 and, the eleventh value, then the probability of a match is 115440/365. If n=482 and, the twelfth value, then the probability of a match is 115921/365. If n=483 and, the thirteenth value, then the probability of a match is 116403/365. If n=484 and, the fourteenth value, then the probability of a match is 116886/365. If n=485 and, the fifteenth value, then the probability of a match is 117370/365. If n=486 and, the sixteenth value, then the probability of a match is 117855/365. If n=487 and, the seventh value, then the probability of a match is 118341/365. If n=488 and, the eighth value, then the probability of a match is 118828/365. If n=489 and, the ninth value, then the probability of a match is 119316/365. If n=490 and, the tenth value, then the probability of a match is 119805/365. If n=491 and, the eleventh value, then the probability of a match is 120295/365. If n=492 and, the twelfth value, then the probability of a match is 120786/365. If n=493 and, the thirteenth value, then the probability of a match is 121278/365. If n=494 and, the fourteenth value, then the probability of a match is 121771/365. If n=495 and, the fifteenth value, then the probability of a match is 122265/365. If n=496 and, the sixteenth value, then the probability of a match is 122760/365. If n=497 and, the seventh value, then the probability of a match is 123256/365. If n=498 and, the eighth value, then the probability of a match is 123753/365. If n=499 and, the ninth value, then the probability of a match is 124251/365. If n=500 and, the tenth value, then the probability of a match is 124750/365. If n=501 and, the eleventh value, then the probability of a match is 125250/365. If n=502 and, the twelfth value, then the probability of a match is 125751/365. If n=503 and, the thirteenth value, then the probability of a match is 126253/365. If n=504 and, the fourteenth value, then the probability of a match is 126756/365. If n=505 and, the fifteenth value, then the probability of a match is 127260/365. If n=506 and, the sixteenth value, then the probability of a match is 127765/365. If n=507 and, the seventh value, then the probability of a match is 128271/365. If n=508 and, the eighth value, then the probability of a match is 128778/365. If n=509 and, the ninth value, then the probability of a match is 129286/365. If n=510 and, the tenth value, then the probability of a match is 129795/365. If n=511 and, the eleventh value, then the probability of a match is 130305/365. If n=512 and, the twelfth value, then the probability of a match is 130816/365. If n=513 and, the thirteenth value, then the probability of a match is 131328/365. If n=514 and, the fourteenth value, then the probability of a match is 131841/365. If n=515 and, the fifteenth value, then the probability of a match is 132355/365. If n=516 and, the sixteenth value, then the probability of a match is 132870/365. If n=517 and, the seventh value, then the probability of a match is 133386/365. If n=518

Thanks to Nico de Vries (100115.2303@compuserve.com) for proposing (on Usenet sci.crypt) a random number generator computer program which he promoted as physically random. Normally, any computer RNG is a deterministic mechanism [20], and simple examination will easily disclose the "amount of internal data" or "state" which it uses. Nico's design was unusual in that it was based on real-time program interactions with IBM PC-type hardware, so it was difficult to know how much internal state there was, or where that state was located. Such a design begged the question of measuring RNG population when only the values themselves were available.

Thanks to Ross Anderson (Ross.Anderson@cl.cam.ac.uk) for suggesting (also on Usenet sci.crypt) that a birthday test could be used to estimate population, and apologies to him also, for my obsession with this idea. Ross sent me email preprints of his birthday test papers [1,2], but then I developed the concept of augmented repetitions, which produced simple equations and exact unique results. All errors in this development are, of course, mine alone.

REFERENCES

1. Anderson, R., R. Gibbons, C. Jagger, F. Kelly and M. Roe. Measuring the Diversity of Random Number Generators. In progress.

2. Anderson, R. Testing Random Number Generators. Submitted for publication in *Electronics Letters*.

3. Boas, M. 1983. *Mathematical Methods in the Physical Sciences*. John Wiley & Sons.

4. Davies, D. and W. Price. 1984. *Security for Computer Networks*. John Wiley & Sons.

5. Goodman, A. and J. Ratti. 1971. *Finite Mathematics with Applications*. Macmillan.

6. Kaliski, B., R. Rivest and A. Sherman. 1985. Is the Data Encryption Standard a Group? (Preliminary Abstract) *Advances in Cryptology -- Eurocrypt '85*. 81-95.

7. Knuth, D. 1973. *The Art of Computer Programming*. Vol. I, *Fundamental Algorithms*. Addison-Wesley.

8. Knuth, D. 1981. *The Art of Computer Programming*. Vol. II, *Seminumerical Algorithms*. Addison-Wesley.

9. Kullback, S. 1976 (first published 1938). *Statistical Methods in Cryptanalysis*. Aegean Park Press, P.O. Box 2837, Laguna Hills, CA 92654-0837.

10. Holtzman, J. 1987. The KEPROM: Sinking the Software Pirates. *Radio-Electronics*. June. 100-104.

11. Letham, L., D. Hoff and A. Folmsbee. 1986. A 128K EPROM Using Encryption of Pseudorandom Numbers to Enable Read Access. *IEEE Journal of Solid-State Circuits*. SC-21(5): 881-888.

12. Marsaglia, G. 1984. A Current View of Random Number Generators. *Proceedings of the Sixteenth Symposium on the Interface Between Computer Science and Statistics*. 3-10.

13. Merkle, R. 1982. *Secrecy, Authentication, and Public Key Systems*. UMI Research Press, University Microfilms International, Ann Arbor, Michigan 48106.

14. Meyer, C. and S. Matyas. 1982. *Cryptography: A New Dimension in Data Security*. John Wiley & Sons.

15. Morita, H., K. Ohta and S. Miyaguchi. 1991. A switching closure test to analyze cryptosystems. *Advances in Cryptology -- CRYPTO '91*. 183-193.

16. Parratt, L. 1961. *Probability and Experimental Errors in Science*. Dover.

17. Patterson, W. 1987. *Mathematical Cryptology*. Rowman & Littlefield, 81 Adams Drive, Totowa, New Jersey 07512.

18. Press, W. et. al. 1986. *Numerical Recipes*. Cambridge University Press.

19. Quisquater, J. and J. Delescaille. 1989. How easy is collision search? Applications to DES. *Advances in Cryptology -- Eurocrypt '89*. 429-434.

20. Ritter, T. 1991. The Efficient Generation of Cryptographic Confusion Sequences. *Cryptologia*. 15(2): 81-139.

21. Seberry, J. and J. Pieprzyk. 1989. *Cryptography: An Introduction to Computer Security*. Prentice Hall.

22. Zollo, S. 1985. IC Keeps Pirates Out. *Electronics Week*. February 11: 73, 76.

APPENDIX A: The sci.crypt Newsgroup

Sci.crypt is the name of a news group on Usenet, a bulletin-board network of computers which use dial-up lines for network communication. Usenet was originally developed using the UUCP (Unix-to-Unix Copy) package distributed with Unix, plus free news software which collects messages in many hundreds of different topics or "groups." On Usenet, messages are passed from computer to computer (until the messages hopefully reach every connected computer), so there is no central facility and no single overall ownership. Electronic mail is supported in the same way.

Access to Usenet News is generally also available on the Internet, an exponentially growing interconnection of networks. Although once confined to higher education and research laboratories, the Internet is increasingly available to ordinary citizens who pay a fee for an account on a computer which is Internet connected.

Those unacquainted with the Internet and News could be disappointed in what they find. Huge numbers of individuals have access to these groups, and many of the groups (like sci.crypt) are unmoderated. Anyone can respond to any message, and much of what is authoritatively expounded is also wrong, or perhaps just incomplete. New rank novices appear almost every day. Not only is it hard to know what to believe, it can be hard simply to find time to read through the chaff (currently tens of messages per day in sci.crypt) to get to the occasional kernel of new information. In addition, the bulletin-board format seems to encourage vituperative responses or "flames," which naturally produce similar counter-responses. These "flame wars" can be disconcerting to those used to respectful intellectual discourse.

At one time sci.crypt supported mostly technical discussions, and these occasionally still appear. But with continuing exponential growth, the advent of an available public key package (Pretty Good Privacy or PGP) and the U.S. government's "Clipper Chip" proposal, most of the traffic in the past year has addressed political issues.

APPENDIX B: The Intel 27916 Keprom

The 27916 Keprom is an integrated circuit memory device designed at Intel Corporation in the early 80's [22]. The Keprom is a "Keyed-Access EPROM" (Erasable Programmable Read-Only Memory) which performs an authentication handshake, during which physically-random values are encrypted under a programmed key [10]. In this way, the data in the EPROM are unlocked only if the associated equipment contains another Keprom with the correct key. Apparently the device was intended to be a technical solution to the problem of software piracy, perhaps especially in the games market. Unfortunately, the device appeared at about the same time that major software manufacturers were dropping copy protection.

The issue of interest here is that the Keprom device included an on-chip physically-random number generator, and a technical paper [11] reported results from evaluation tests on that generator. Briefly, the design consists of a 32-bit shift-register which accumulates bits sampled from a drifting on-chip oscillator, noise, and feedback from the register itself. Because 32-bit values are reported from the generator, we expect to estimate a 32-bit population. However, the higher-order repetitions in the reported data imply a value very much less than this, which may indicate problems in the physically-random design.

LARGE TABLES

Table 4.1 All Possible Trials for N=2 and s=6

Trial	Repetitions	Aug. Reps.	Accum. Aug. Reps.
0 0 0 0 0 0	1 0 0 0 0	1 6 15 20 15	1 6 15 20 15
0 0 0 0 0 1	0 1 0 0 0	0 1 5 10 10	1 7 20 30 25
0 0 0 0 1 0	0 1 0 0 0	0 1 5 10 10	1 8 25 40 35
0 0 0 0 1 1	0 0 1 0 1	0 0 1 4 7	1 8 26 44 42
0 0 0 1 0 0	0 1 0 0 0	0 1 5 10 10	1 9 31 54 52
0 0 0 1 0 1	0 0 1 0 1	0 0 1 4 7	1 9 32 58 59
0 0 0 1 1 0	0 0 1 0 1	0 0 1 4 7	1 9 33 62 66
0 0 0 1 1 1	0 0 0 2 0	0 0 0 2 6	1 9 33 64 72
0 0 1 0 0 0	0 1 0 0 0	0 1 5 10 10	1 10 38 74 82
0 0 1 0 0 1	0 0 1 0 1	0 0 1 4 7	1 10 39 78 89
0 0 1 0 1 0	0 0 1 0 1	0 0 1 4 7	1 10 40 82 96
0 0 1 0 1 1	0 0 0 2 0	0 0 0 2 6	1 10 40 84 102
0 0 1 1 0 0	0 0 1 0 1	0 0 1 4 7	1 10 41 88 109
0 0 1 1 0 1	0 0 0 2 0	0 0 0 2 6	1 10 41 90 115
0 0 1 1 1 0	0 0 0 2 0	0 0 0 2 6	1 10 41 92 121
0 0 1 1 1 1	0 0 1 0 1	0 0 1 4 7	1 10 42 96 128
0 1 0 0 0 0	0 1 0 0 0	0 1 5 10 10	1 11 47 106 138
0 1 0 0 0 1	0 0 1 0 1	0 0 1 4 7	1 11 48 110 145
0 1 0 0 1 0	0 0 1 0 1	0 0 1 4 7	1 11 49 114 152
0 1 0 0 1 1	0 0 0 2 0	0 0 0 2 6	1 11 49 116 158
0 1 0 1 0 0	0 0 1 0 1	0 0 1 4 7	1 11 50 120 165
0 1 0 1 0 1	0 0 0 2 0	0 0 0 2 6	1 11 50 122 171
0 1 0 1 1 0	0 0 0 2 0	0 0 0 2 6	1 11 50 124 177
0 1 0 1 1 1	0 0 1 0 1	0 0 1 4 7	1 11 51 128 184
0 1 1 0 0 0	0 0 1 0 1	0 0 1 4 7	1 11 52 132 191
0 1 1 0 0 1	0 0 0 2 0	0 0 0 2 6	1 11 52 134 197
0 1 1 0 1 0	0 0 0 2 0	0 0 0 2 6	1 11 52 136 203
0 1 1 0 1 1	0 0 1 0 1	0 0 1 4 7	1 11 53 140 210
0 1 1 1 0 0	0 0 0 2 0	0 0 0 2 6	1 11 53 142 216
0 1 1 1 0 1	0 0 1 0 1	0 0 1 4 7	1 11 54 146 223
0 1 1 1 1 0	0 0 1 0 1	0 0 1 4 7	1 11 55 150 230
0 1 1 1 1 1	0 1 0 0 0	0 1 5 10 10	1 12 60 160 240
1 0 0 0 0 0	0 1 0 0 0	0 1 5 10 10	1 13 65 170 250
1 0 0 0 0 1	0 0 1 0 1	0 0 1 4 7	1 13 66 174 257
1 0 0 0 1 0	0 0 1 0 1	0 0 1 4 7	1 13 67 178 264
1 0 0 0 1 1	0 0 0 2 0	0 0 0 2 6	1 13 67 180 270
1 0 0 1 0 0	0 0 1 0 1	0 0 1 4 7	1 13 68 184 277
1 0 0 1 0 1	0 0 0 2 0	0 0 0 2 6	1 13 68 186 283
1 0 0 1 1 0	0 0 0 2 0	0 0 0 2 6	1 13 68 188 289
1 0 0 1 1 1	0 0 1 0 1	0 0 1 4 7	1 13 69 192 296
1 0 1 0 0 0	0 0 1 0 1	0 0 1 4 7	1 13 70 196 303
1 0 1 0 0 1	0 0 0 2 0	0 0 0 2 6	1 13 70 198 309
1 0 1 0 1 0	0 0 0 2 0	0 0 0 2 6	1 13 70 200 315
1 0 1 0 1 1	0 0 1 0 1	0 0 1 4 7	1 13 71 204 322
1 0 1 1 0 0	0 0 0 2 0	0 0 0 2 6	1 13 71 206 328
1 0 1 1 0 1	0 0 1 0 1	0 0 1 4 7	1 13 72 210 335
1 0 1 1 1 0	0 0 1 0 1	0 0 1 4 7	1 13 73 214 342
1 0 1 1 1 1	0 1 0 0 0	0 1 5 10 10	1 14 78 224 352
1 1 0 0 0 0	0 0 1 0 1	0 0 1 4 7	1 14 79 228 359
1 1 0 0 0 1	0 0 0 2 0	0 0 0 2 6	1 14 79 230 365
1 1 0 0 1 0	0 0 0 2 0	0 0 0 2 6	1 14 79 232 371
1 1 0 0 1 1	0 0 1 0 1	0 0 1 4 7	1 14 80 236 378
1 1 0 1 0 0	0 0 0 2 0	0 0 0 2 6	1 14 80 238 384
1 1 0 1 0 1	0 0 1 0 1	0 0 1 4 7	1 14 81 242 391
1 1 0 1 1 0	0 0 1 0 1	0 0 1 4 7	1 14 82 246 398
1 1 0 1 1 1	0 1 0 0 0	0 1 5 10 10	1 15 87 256 408
1 1 1 0 0 0	0 0 0 2 0	0 0 0 2 6	1 15 87 258 414
1 1 1 0 0 1	0 0 1 0 1	0 0 1 4 7	1 15 88 262 421
1 1 1 0 1 0	0 0 1 0 1	0 0 1 4 7	1 15 89 266 428
1 1 1 0 1 1	0 1 0 0 0	0 1 5 10 10	1 16 94 276 438
1 1 1 1 0 0	0 0 1 0 1	0 0 1 4 7	1 16 95 280 445
1 1 1 1 0 1	0 1 0 0 0	0 1 5 10 10	1 17 100 290 455
1 1 1 1 1 0	0 1 0 0 0	0 1 5 10 10	1 18 105 300 465
1 1 1 1 1 1	1 0 0 0 0	1 6 15 20 15	2 24 120 320 480

For all 64 possible trials of 6 samples from a population of 2 symbols:

XctTot:	2	12	30	40	30
XctEr:	2	12	30	40	30
AugTot:	2	24	120	320	480
AugEar:	2	24	120	320	480

Table 4.2 All Possible Trials for N=3 and s=4

Trial	Reps	Aug. Reps.	Accum. Aug. Reps.
0 0 0 0	1 0 0	1 4 6	1 4 6
0 0 0 1	0 1 0	0 1 3	1 5 9
0 0 0 2	0 1 0	0 1 3	1 6 12
0 0 1 0	0 1 0	0 1 3	1 7 15
0 0 1 1	0 0 2	0 0 2	1 7 17
0 0 1 2	0 0 1	0 0 1	1 7 18
0 0 2 0	0 1 0	0 1 3	1 8 21
0 0 2 1	0 0 1	0 0 1	1 8 22
0 0 2 2	0 0 2	0 0 2	1 8 24
0 1 0 0	0 1 0	0 1 3	1 9 27
0 1 0 1	0 0 2	0 0 2	1 9 29
0 1 0 2	0 0 1	0 0 1	1 9 30
0 1 1 0	0 0 2	0 0 2	1 9 32
0 1 1 1	0 1 0	0 1 3	1 10 35
0 1 1 2	0 0 1	0 0 1	1 10 36
0 1 2 0	0 0 1	0 0 1	1 10 37
0 1 2 1	0 0 1	0 0 1	1 10 38
0 1 2 2	0 0 1	0 0 1	1 10 39
0 2 0 0	0 1 0	0 1 3	1 11 42
0 2 0 1	0 0 1	0 0 1	1 11 43
0 2 0 2	0 0 2	0 0 2	1 11 45
0 2 1 0	0 0 1	0 0 1	1 11 46
0 2 1 1	0 0 1	0 0 1	1 11 47
0 2 1 2	0 0 1	0 0 1	1 11 48
0 2 2 0	0 0 2	0 0 2	1 11 50
0 2 2 1	0 0 1	0 0 1	1 11 51
0 2 2 2	0 1 0	0 1 3	1 12 54
1 0 0 0	0 1 0	0 1 3	1 13 57
1 0 0 1	0 0 2	0 0 2	1 13 59
1 0 0 2	0 0 1	0 0 1	1 13 60
1 0 1 0	0 0 2	0 0 2	1 13 62
1 0 1 1	0 1 0	0 1 3	1 14 65
1 0 1 2	0 0 1	0 0 1	1 14 66
1 0 2 0	0 0 1	0 0 1	1 14 67
1 0 2 1	0 0 1	0 0 1	1 14 68
1 0 2 2	0 0 1	0 0 1	1 14 69
1 1 0 0	0 0 2	0 0 2	1 14 71
1 1 0 1	0 1 0	0 1 3	1 15 74
1 1 0 2	0 0 1	0 0 1	1 15 75
1 1 1 0	0 1 0	0 1 3	1 16 78
1 1 1 1	1 0 0	1 4 6	2 20 84
1 1 1 2	0 1 0	0 1 3	2 21 87
1 1 2 0	0 0 1	0 0 1	2 21 88
1 1 2 1	0 1 0	0 1 3	2 22 91
1 1 2 2	0 0 2	0 0 2	2 22 93
1 2 0 0	0 0 1	0 0 1	2 22 94
1 2 0 1	0 0 1	0 0 1	2 22 95
1 2 0 2	0 0 1	0 0 1	2 22 96
1 2 1 0	0 0 1	0 0 1	2 22 97
1 2 1 1	0 1 0	0 1 3	2 23 100
1 2 1 2	0 0 2	0 0 2	2 23 102
1 2 2 0	0 0 1	0 0 1	2 23 103
1 2 2 1	0 0 2	0 0 2	2 23 105
1 2 2 2	0 1 0	0 1 3	2 24 108
2 0 0 0	0 1 0	0 1 3	2 25 111
2 0 0 1	0 0 1	0 0 1	2 25 112
2 0 0 2	0 0 2	0 0 2	2 25 114
2 0 1 0	0 0 1	0 0 1	2 25 115
2 0 1 1	0 0 1	0 0 1	2 25 116
2 0 1 2	0 0 1	0 0 1	2 25 117
2 0 2 0	0 0 2	0 0 2	2 25 119
2 0 2 1	0 0 1	0 0 1	2 25 120
2 0 2 2	0 1 0	0 1 3	2 26 123
2 1 0 0	0 0 1	0 0 1	2 26 124
2 1 0 1	0 0 1	0 0 1	2 26 125
2 1 0 2	0 0 1	0 0 1	2 26 126
2 1 1 0	0 0 1	0 0 1	2 26 127
2 1 1 1	0 1 0	0 1 3	2 27 130
2 1 1 2	0 0 2	0 0 2	2 27 132
2 1 2 0	0 0 1	0 0 1	2 27 133
2 1 2 1	0 0 2	0 0 2	2 27 135
2 1 2 2	0 1 0	0 1 3	2 28 138
2 2 0 0	0 0 2	0 0 2	2 28 140
2 2 0 1	0 0 1	0 0 1	2 28 141
2 2 0 2	0 1 0	0 1 3	2 29 144
2 2 1 0	0 0 1	0 0 1	2 29 145
2 2 1 1	0 0 2	0 0 2	2 29 147
2 2 1 2	0 1 0	0 1 3	2 30 150
2 2 2 0	0 1 0	0 1 3	2 31 153
2 2 2 1	0 1 0	0 1 3	2 32 156
2 2 2 2	1 0 0	1 4 6	3 36 162

For all 81 possible trials of 4 samples from a population of 3 symbols:

XctTot:	3	24	72
XctEr:	3	24	72
AugTot:	3	36	162
AugEar:	3	36	162

Table 4.3 Summary of All Possible Trials for N=10 s=3 and N=100 s=2

For all 1000 possible trials of 3 samples from a population of 10 symbols:

XctTot:	10	270
XctEr:	10	270
AugTot:	10	300
AugEar:	10	300

For all 10000 possible trials of 2 samples from a population of 100 symbols:

XctTot:	100
XctEr:	100
AugTot:	100
AugEar:	100

BIOGRAPHICAL SKETCH

Terry Ritter is a registered Professional Engineer who has spent the last five years conducting independent research and development on cryptographic systems. Mr. Ritter is a member of the IEEE and ACM, and holds the US patent on Dynamic Substitution technology; he has previously contributed articles on reversible nonlinear cryptographic combiners and a survey of cryptographic random number generators. Ritter Software Engineering offers fast cipher engines to hardware and software developers.

Terry Ritter, his *current address*, and his *top page*.

Last updated: 1998-02-24

Population Estimation Worksheets in JavaScript

Using Augmented 2-Reps

A *Ciphers By Ritter* Page

Terry Ritter

Suppose we have a random-number machine for use in a game of chance. The machine has an opaque covering, and contains some number of tokens which the machine somehow physically mixes. The tokens can be identified by serial number, and each is replaced before another is drawn. Is there a way to externally check the honesty of the system?

If we collect samples until we start seeing tokens appear multiple times, then repeat this number of samples for a number of different trials, we can make some statistical inferences. The relationship between the size of the population and the expected number of repetitions is discussed in my 1994 article [1], Estimating Population from Repetitions in Accumulated Random Samples (82K).

Augmented Repetitions

If we get some token twice (in a single trial), we have a "double" or a "2-rep," and if some token occurs three times, we have a "triple" or a "3-rep." It turns out that one 3-rep has the same probability of occurring as three 2-reps. We can convert each "n-rep" to a count of C(n,2) [that is, the number of combinations of *n* things taken 2 at a time] 2-reps. When all "n-reps" have been converted to 2-reps and summed, we get a value for *augmented* 2-reps which is easily related to population, assuming that all tokens have an equal probability. If some tokens appear more frequently than expected (on average), we will get more augmented repetitions, and predict a smaller than expected value for the "effective" population.

The Worksheets

The first worksheet is used to enter the assumed population, which is used to compute a reasonable sample size. The computed sample size should produce 4 augmented 2-reps, on average, provided each element has an equal probability. Any significant number of non-equal probabilities will raise the average number of augmented doubles found (over a number of experiments).

The first worksheet is *also* used to enter the actual sample size, which is used to compute the theoretically-expected count for each repetition level. The predicted counts are displayed on the second worksheet as benchmarks to which we can compare the counts found by experiment. This comparison will highlight problems in the *distribution* of n-reps.

The second worksheet performs the computations necessary to find the equivalent number of "augmented 2-reps." The user conducts a substantial number of same-size trials on the population, and develops an average count for the number of occurrences of each n-rep. (In one trial, if the value 17 occurs three times, and the value 5 also occurs three times, we have two 3-reps, for an occurrence count of 2 at the 3-reps level.) By entering the (generally fractional) average occurrence counts at each repetition level, the equivalent number of augmented 2-reps is accumulated as the final value in the rightmost or "Running Total 2-Reps" column.

The third worksheet estimates the population from the entered trial size and a total of augmented 2-reps. In general, finding twice the number of 2-reps predicts a population of half the size or a one bit smaller "address space."

For Predicting Repetition Counts		
Enter Assumed Population	Reasonable Sample Size	**Enter** Actual Sample Size

Augmented Repetition Computations				
Rep. Level	Predicted Count	**Enter** Actual Count	Equivalent Count in 2-Reps	Running Total Aug. 2-Reps

Population Estimation Computations			
Enter Sample Size	**Enter** Avg. Aug. 2-Reps	Est. Population	Est. Pop. in Bits

Keprom Data

In the early 80's, Intel Corporation designed an IC they called a "Keprom," or "Keyed-Access EPROM." The Keprom device had an on-chip physically-random number generator, and a technical paper [2] reported results from tests on that generator. Briefly, the design consists of a 32-bit shift-register which accumulates bits sampled from a drifting on-chip oscillator, noise, and feedback from the register itself.

Clicking on the "Keprom" button introduces the measured values into the worksheets. While the resulting 30-bit population value may seem "not too different" from the expected 32 bits, this is only 1/4 the expected size. That is, in this design, 3 out of every 4 of the possible values might as well not even exist.

The Keprom experiment results also show quite a few repetitions above the 4-rep level. Such higher-level repetitions should be very improbable in a true random system of the indicated size. Accordingly, the experimental results would warrant an investigation into the actual values involved and how these particular values could be so vastly more probable than one would expect. For example, we should expect to find about 4×10^{-73} 18-reps on average (that is, we should never actually see one), given the assumed Keprom population and sample size. This makes the 18-rep which was found experimentally extremely unlikely to have been produced by a flat distribution.

References

[1] Ritter, T. 1994. Estimating Population from Repetitions in Accumulated Random Samples. *Cryptologia.* 18(2):155-190.

[2] Letham, L., D. Hoff and A. Folmsbee. 1986. A 128K EPROM Using Encryption of Pseudorandom Numbers to Enable Read Access. *IEEE Journal of Solid-State Circuits.* SC-21(5): 881-888.

***Terry Ritter*, his *current address*, and his *top page*.**

***Last updated:*1996-09-20**

PUBLISHED: Ritter, T. 1992. Voice and Video Cryptography in a DSP Environment. Presented at, and published in, *The Proceedings of The Second Annual Texas Instruments TMS320 Educators Conference*, August 5-7 1992, in Houston, Texas.

Voice and Video Cryptography in a DSP Environment

Terry Ritter

ABSTRACT: A quick introduction to cryptography leading up to various DSP product possibilities.

Are your customers concerned about information security? If not, they probably should be. Eavesdropping on phone extensions certainly seems common in the movies. Kids with scanners really do listen in on cellular phone conversations for fun; others may have their own reasons. Some wire tapping may occur which is completely outside that authorized by the courts. And there are times in which even honest individuals would like to communicate in absolute privacy.

CRYPTOGRAPHY

Cryptography is the art of using a secret to confuse a message so that it may be communicated without being understood except by those who know the secret. We call that secret the "key," because it can be used to "lock" and "unlock" messages. Using cryptography is much like sealing the envelope of a first-class letter, except that nobody can open a cryptographic letter in transit.

Non-enciphered information is called "plaintext" until it is enciphered, then we have "ciphertext." We assume the potential existence of an "opponent" who may someday try to recover the information in our ciphertext by "cryptanalysis," or "cracking" the cipher. We will assume that the opponent will know everything about our equipment except the secret keys; the opponent may even know some of the plaintext we send, and may use this to try and identify our key for later use. "Strong" ciphers can remain "secure" or unbroken despite serious and well-financed technical "attacks." But it is not always necessary to "break" a cipher to penetrate security.

Cryptography and The Government

Because the government has traditionally controlled cryptographic technology (especially since WWII), there is currently an argument about whether the public availability of cryptographic equipment will upset the balance between "cops" and "robbers." To the extent that the cops depend upon wiretaps for information, there may be something to this. But cryptography does not obscure income without a source, does not hide the shipment or sale of illegal goods, and has no effect on victims, witnesses, or spies. In fact, about the most cryptography can do is to return things to the way they were when the Constitution was written, when there were no wires to tap, and no radio technology on which to hear private conversations.

The few former officials who comment about government policy on cryptography sometimes say that even minimal ciphers will ensure personal privacy, so that serious cryptography would be overkill for ordinary citizens. But then the question arises: Privacy from whom? If one wishes to keep a secret from the guy down the hall, that is one thing. But keeping a secret from a serious detective agency or a motivated computer "cracker" is quite another. The simple facts are that there are many people who could break simple ciphers, they have lots of equipment to help them, and occasionally these people may be given reason to act.

Some texts claim that the way to tell whether one needs cryptography is to look to what one has to protect, and how much that protection is worth. But modern cryptography need not be expensive, so discussion of worth is really beside the point. Soon, every digital phone, every modem, and every fax machine will have scant reason not to be able to seal the envelope of communication.

Ciphers used by ordinary individuals should be designed to be uncrackable, even with world-class computing resources, because any other solution is just unworkable. If we manage to define a cipher that only the government can crack (only under court order, of course), then, five or ten years later, many governments, corporations, and perhaps even serious criminal enterprises will be able to do the same thing. And if we give all our keys to the government, we can be sure that some of those keys will end up where they are not supposed to be.

Cryptography and National Security

Modern cryptography is a computer-based technology, and every kid with a computer and a cipher program can participate. This means that cryptography is already beyond being a National Security issue: Certainly any country has (or could have) many secure cipher designs to choose from. Indeed, some of the best public work in cryptography is coming into the U.S. from Europe, Israel and China, so the science of cryptography is hardly ours to conceal.

Criminals with technical talent (or the resources to buy such talent) can make and use serious cryptography whenever they think it worthwhile. This means that cryptography is beyond being a Crime Fighting issue too, except for unsophisticated crooks. But it is a fearful cost to prevent the use of cryptography by business and private individuals simply on the hope of someday wire-tapping small-time criminal organizations. It would make more sense for law enforcement to encourage the public use of cryptography to address crimes of information theft.

Like most technologies, cryptography may be used for good or evil, but it will be used. I suggest that its best use is in the constitutional protection of privacy and free speech by and for the individual.

INFORMATION SECURITY

Cryptography is only one part of information security, and the creation and management of a secure information environment can be extremely difficult. If users save paper copies of sensitive information, then "acquiring" such a copy would be far easier than any direct technical attack on a cipher. Employees in such an environment can be at risk of temptation, compromise or intimidation. The environment itself could be penetrated by guile or surreptitious entry. And the trash could be examined for documents which were not properly burned or shredded.

If users put sensitive plaintext on disk, such disks also must be kept in a secure area. Since it is not easy to know whether a disk holds secure information, disks cannot be allowed to flow in and out. Portable computers probably should not be allowed in a secure area unless they remain closed or use a comprehensive cryptographic system. Since CRT displays can radiate video information for a block or so, secure machines will use LCD or other low-RF-emission displays.

If a cryptographic key has been compromised, we want to know this so that we can have the keys changed. But usually we will not know when a key is lost, so, to limit damage, we will have to change keys periodically, whether we like it or not. And if someone can "acquire" sensitive information without penetrating the cipher, that information can be used in an effort to get the cipher changed to something less effective. Thus, constant attempts to be aware of any disclosure, and to identify the exact avenue of such disclosure, are yet other security management issues (as well as the basis for a lot of spy intrigue).

Fortunately, personal voice communications are another matter. Unless conversations are recorded and then held insecurely, they are transient, and only exist as "plaintext" while they are occurring. This means that locations for secure conversations need only be transiently secure; we might take a walk around the neighborhood, for example. Of course, we still may worry about electronic "bugs" in offices and rooms. And if someone lets a secret slip after a conversation, well, that is part of security too. But it is not something that cryptography can do anything about.

CIPHER DESIGNS

Talking about the "best" cipher design is something like talking about the "best" computer design. There are many different approaches, and also valid reasons why technically superior designs may not be better.

Because the same cipher must be present at both ends of a conversation, cryptography seems to demand standards. Currently, there are few good standards, possibly due to government opposition. But it is less necessary to have a standard if you have one of the first products available in the right price range.

There are few absolute assurances in cipher design; cryptography may be a product which cannot be guaranteed. During WWII, many apparently "strong" ciphers were broken by direct technical attack. Modern computer ciphers can be far, far more complex than WWII ciphers, but there still are no guarantees. In most cases, it will be impossible to prove that a particular cipher design cannot be broken, or even to guarantee some minimum amount of effort will be required to break it. While we can try to predict the work needed to crack a cipher under the best known attack, we cannot reason about attacks which we do not know. In any case, it is not necessary to "break" a cipher in order to penetrate security.

In analyzing the "strength" of a cryptographic system, we generally must assume that the opponent will know everything about the equipment except the key. Moreover, even if the opponent knows the plaintext of a message, we still expect the design to hide the key. These requirements can be surprisingly difficult to meet.

Keys

Perhaps the most awkward aspect of cryptography is the need for keys. In a classic single key cipher, users must have the same key on both ends, cannot communicate keys "in the clear," and each end must hold their key in absolute secrecy. In effect, the secrecy of the entire system rests on the secrecy of the "key," and that key exists in two locations. A momentary lapse in either place could compromise all future communications under that key without breaking the cipher.

In a secret key design, it can be difficult to distribute keys to distant locations. And secure distribution can be effectively impossible within a short time frame, such as when the current key has been exposed and so must be changed right now. But when secret key distribution can be made to work, a personally-delivered key carries an inherent validity which is hard to deny.

A newer approach, still under U.S. patent, is called "public key" technology. Many people eventually find this name confusing, because this scheme, too, relies upon secrecy. Public key is a two- key technology, in which only one of the keys is public; the other (associated) key still must be held in absolute secrecy. One of the key values can be used to encipher a message, but it will take the associated keyja different valuejto decipher the message. Thus, we can afford to send the "public" key "in the clear," and we only need to protect one location.

Some public key techniques also provide the capability to use a secret key to "sign" a message which anyone who has the associated public key can verify. Thus, various sorts of "certificates" can be issued, including testimony to the validity of particular public key values. Unfortunately, this method of key distribution can imply the creation and maintenance of a bureaucracy of "trusted certification authorities."

If we have a direct end-to-end phone connection, public key technology is great. In such a context, there should be no possibility that anybody could be "in the middle" to delete "key change" messages, re-write the key mentioned in such messages, or convince each end to use a new key. With a valid end-to-end connection, if the far end changes their keys, they just give you their public key "in the clear," and go on.

Unfortunately, it is not always possible to assure that we are not communicating through some high-performance "middleman." In this context, we might seek to use the public key validation feature to certify a new key, but this depends upon having a good key pair in the other direction. This is not possible when we start out, nor if both ends have changed their keys before the current contact. Especially in the context of a store-and-forward computer communications network, many tricks are possible, even with public keys. By pretending to be someone else, a middleman process might be able to convert both ends to known keys and thus penetrate security even though the cipher itself remains technically unbroken.

Public key techniques generally make use of multiplication, exponentiation and division on very large numbers (250 or more decimal digits). Consequently, public key techniques are relatively slow, and they are generally used simply to transfer a secret key. That secret key is then used in a conventional one-key cipher for actually communicating data. Obviously, two-key techniques have not made one-key designs obsolete.

Different Types of Ciphers

A classic distinction between ciphers is whether they encipher symbols one-by-one, or require some accumulation before enciphering. Accumulating symbols in blocks will imply delay, unless the data are inherently produced in the correct-size blocks. In contrast, stream ciphers encipher each symbol separately, implying a minimum of delay in the cipher process.

Some researchers insist that the defining difference between block and stream ciphers should be that block ciphers always encipher a particular block in a particular way, while stream ciphers encipher subsequent elements differently. Unfortunately, this definition makes Simple Substitution on 8-bit characters a "block" cipher, while my Dynamic Transposition cipher on 4096-bit blocks becomes a "stream" cipher. Other authors distinguish between block and stream ciphers on the basis of whether a ciphertext error is "expanded" by deciphering. But modern variations can be expanding, limited-expanding, or non-expanding in either stream or block formats. Such confusion seems to be the natural consequence of folding new concepts into old definitions.

Another distinction between designs is that ciphers may need to work in an imperfect environment. Errors in the transmitted ciphertext will naturally produce errors in the deciphered plaintext, but in many ciphers, a single transmission error will cause many plaintext errors; this is "error expansion." In voice communication, error expansion can have the effect of moving the connection quality from "passable" to "unintelligible." A partial defense would be to add forward error correction to the ciphertext bit stream. Of course, computer communications which are not time- sensitive can always be made essentially error-free by using automatic error-detection and re-transmission schemes.

Ciphers which do not cause error expansion have their own problems. Typically these create an internal bit-stream (or "running key") which must be kept synchronized to a similar stream at the far end. Transmission errors may be non-expanded if sync is maintained, but if sync is lost, this will have to be detected, and some higher-level process will have to recover it. This could imply a periodic transmission of cipher sync information, or a real-time negotiated re-initialization, and these needs may be more of an imposition than simply dealing with error expansion.

Voice Ciphers

In the case of voice ciphers (generally called "scramblers"), one obvious distinction is between analog and digital cipher technology. It turns out that secure analog scrambling is surprisingly difficult. First, it is necessary to maintain a similar frequency range and energy distribution to allow the ciphertext to be transported over voice channels. Next, the human ear and brain is amazingly adept at hearing words in recorded scrambled conversation, despite extensive processing in both the frequency and time domains. Consequently, good scramblers are usually digital.

Because good voice cryptography works on a digital data stream, it depends upon a good vocoder design and a good modem design. A cipher also places requirements on, and is affected by, the signalling format or "data frame" used for communication, so the system design is not quite as simple as just connecting boxes together. In DSP, the modem, vocoder, and cipher will all be functioning in real time, and this will place serious requirements on the DSP implementation itself.

Special requirements for telephone cryptography require operation in an environment of satellite communication delays, and possibly even conference calls. While it is easy to make a system which will decipher a single communication in several different places, it is more difficult to allow different parties to exchange "cipher origination" status in real time while maintaining security. Such conversations may be essentially half-duplex, although this would be automatic.

CRYPTOGRAPHY AND DIGITAL SIGNAL PROCESSING

When we need digital processing of analog signals we naturally turn to DSP. DSP products with cipher technology might include:

- Secure Digital Cellular Phones,
- Secure Business Phone Systems,
- Secure Home Phones,
- Secure Data Modems,
- Secure Voice and Data Modems,
- Secure Video Conferencing Modems,
- Secure Data and Fax Modems,
- Secure Fax Machines, and
- Secure Digital Audio Recorders.

The Secure Phone

A crypto voice instrument could be constructed which would generate new key-pairs for its users, who could change keys at any time. After assuring that the other party really is who they are supposed to be, the user could direct a public key to be sent to, or accepted from, the far end. The associated private keys would be stored internally, and never displayed, transmitted or otherwise revealed. Since nobody could read such a key and take it away for use (or sale), primary security would reduce to restricting access to the instrument itself. This would seem to be a very useful and flexible approach, both for home and business use. High security might imply guarded sign-in access to exactly the same sort of instrument. New security requirements might imply new keys and increased attention to physical security, but would require no new equipment.

More advanced designs could automatically negotiate public key transfer (but only if just one direction needed a new key). They also might recognize multiple encryption standards (provided each standard was designed to be recognized). In effect, the secure phone might operate almost exactly like a normal phone most of the time, thus requiring a minimum of training for proper use. Similar comments could be made about secure data modems and secure fax machines.

New voice network services can have a major effect on cryptographic ease-of-use and security. For example, if we have the ability to identify the number of a caller, we can use this to automatically identify a corresponding crypto key (if any). Thus, once the system is set up, there need be no frantic moments of remembering the correct key, or consulting key diaries (which must then be held in absolute secrecy). In fact, we need not even re- negotiate keys until we need new ones, and so we minimize the possibility that a hidden "middleman" process could deceive us.

"Really" Random Numbers

One feature frequently needed by serious ciphers is the ability to produce "really" random numbers. Now, various sorts of Random Number Generator (RNG) designs are used in many ciphers, but here we refer to "really" random numbers. No deterministic machine can produce "really" random numbers, so no program or algorithm can be much help. What we need is some sort of on-chip physically-random source, such as amplified diode noise, dark-current capacitor leakage, etc. Some designs have tried to use multiple deliberately- drifting asynchronous oscillators only to find that the results were not really very random after all.

"Really" random numbers are important, because they are easy to protect in a very strong way. We can send "really" random numbers as short "messages." Since any such "message" is valid, a particular message extends no statistical lever for use in breaking the cipher. This means that we can easily send a random value to the far end using our fixed key, and then use the deciphered message -- a completely random value -- as the key for the actual conversation. These values are known as "message keys," and they prevent us from using the same key over and over, except on "really" random values which do not support analysis.

SUMMARY

In the past, good cryptography has been expensive, hard to get and difficult to use. In the future, cryptography will be mostly cheap, widely available and automatic. We have a choice: We can sit back and let the rest of the world develop this market, or we can work at developing it ourselves, by setting standards and providing high quality products at reasonable prices. We can do this now in our own country, while encouraging Congress to allow our equipment to be exported as soon as possible.

GENERAL BIBLIOGRAPHY

The classic historical introduction to the field is Kahn [10], and current governmental issues are discussed in Barlow [1] and Murray [14]. Beker and Piper [2,3] discuss actual equipment design issues, but these are now somewhat dated; Pfleeger [16] and Seberry and Pieprzyk [22] are newer but more professorial. Deavours and others [5] covers a range of real issues in articles reprinted from Cryptologia, and Graff and Sheets [9] gives some information on real world scrambling systems. Ciarcia [4] and Pearson [15] are an excellent example of how tricky the field is; first study Ciarcia (a real circuit design), and only then read Pearson (how the design is broken). Geffe [8] and Siegenthaler [23] provide a more technical lesson, and Kochanski [11,12] cracks some common PC cipher programs. My own work is included mainly to indicate my background for this paper.

References

1. Barlow, J. 1992. Decrypting the Puzzle Palace. *Communications of the ACM.* 35(7): 25-31.

2. Beker, H. and F. Piper. 1982. *Cipher Systems.* Wiley.

3. Beker, H. and F. Piper. 1985. *Secure Speech Communications.* Academic Press.

4. Ciarcia, S. 1986. Build a Hardware Data Encryptor. *Byte.* September. 97-111.

5. Deavours, C., D. Kahn, L. Kruh, G. Mellen, and B. Winkel (Eds.) 1987. *Cryptology Yesterday, Today, and Tomorrow.* Artech House.

6. Denning, D. 1982. *Cryptography and Data Security.* Addison-Wesley.

7. Ding, C, G. Xiao and W. Shan. 1991. *The Stability of Stream Ciphers.* Springer-Verlag. Lecture Notes in Computer Science #561.

8. Geffe, P. 1973. How to protect data with ciphers that are really hard to break. *Electronics.* January 4. 99-101.

9. Graff, R. and W. Sheets. 1987. *Video Scrambling & Descramblng for Satellite & Cable TV.* Howard W. Sams.

10. Kahn, D. 1967. *The Codebreakers.* Macmillan.

11. Kochanski, M. 1987. A Survey of Data Insecurity Packages. *Cryptologia.* 11(1): 1-15.

12. Kochanski, M. 1988. Another Data Insecurity Package. *Cryptologia.* 12(3): 165-173.

13. Meyer, C. and S. Matyas. 1982. *Cryptography: A New Dimension in Data Security.* John Wiley & Sons.

14. Murray, W. 1992. Who Holds the Keys? *Communications of the ACM.* 35(7): 13-15.

15. Pearson, P. 1988. Cryptanalysis of the Ciarcia Circuit Cellar Data Encryptor. *Cryptologia.* 12(1): 1-9.

16. Pfleeger, C. 1989. *Security in Computing.* Prentice Hall.

17. Rueppel, R. 1986. *Analysis and Design of Stream Ciphers.* Springer-Verlag.

18. Ritter, T. 1990. Substitution Cipher with Pseudo-Random Shuffling: The Dynamic Substitution Combiner. *Cryptologia.* 14(4): 289-303.

19. Ritter, T. 1991. Dynamic Substitution Combiner and Extractor. *US Patent 4,979,832*, issued Dec. 25, 1990.

20. Ritter, T. 1991. Transposition Cipher with Pseudo-Random Shuffling: The Dynamic Transposition Combiner. *Cryptologia.* 15(1): 1-17.

21. Ritter, T. 1991. The Efficient Generation of Cryptographic Confusion Sequences. *Cryptologia.* 15(2): 81-139.

22. Seberry, J. and J. Pieprzyk. 1989. *Cryptography: An Introduction to Computer Security.* Prentice Hall.

23. Siegenthaler, T. 1985. Decrypting a Class of Stream Ciphers Using Ciphertext Only. *IEEE Transactions on Computers.* C-34: 81-85.

24. van Eck, W. 1985. Electromagnetic Radiation from Video Display Units: An Eavesdropping Risk? *Computers & Security.* 4: 269-286.

Note: Cryptologia was previously published by Rose-Hulman Institute of Technology. As of 1995-07, Cryptologia is published by the Department of Mathematical Sciences, United States Military Academy, West Point NY 10996-9902 USA. Subscriptions are $40 per year in the US, and many back issues are available at $11 each.

ABOUT THE AUTHOR

Terry Ritter is a Registered Professional Engineer, and a member of IEEE and ACM. Ritter Software Engineering offers research, development and design services in cryptography, software, computer architecture and other fields.

Terry Ritter, his *current address*, and his *top page*.

Last updated: 1995-11-11

The Efficient Generation of Cryptographic Confusion Sequences

Terry Ritter

Cryptologia, Volume 7, Number 4, October 1983, pp. 305-339.
Copyright © 1983 Terry Ritter

An invertible or keyed sequence of numerical, 16-bit, pseudo-random values is used to control substitution through a large, dynamic, table. Such a sequence can be generated by a cryptographic stream cipher, such as a stream cipher based on a Fibonacci or Galois shift register, or a block cipher in a counter or other mode. Such a sequence can also be generated by a pair of fast, cryptographic, substitution tables. Such tables can be quickly generated from a key using a cryptographic sequence. Such tables can be used to generate a very long, very complex, sequence of 16-bit values. Such a sequence can be used to control a dynamic substitution table, which can then be used to encipher and decipher a message. Such a system can be very fast, very secure, and very complex. Such a system can be made to depend on any cryptographic primitive, with the generation of tables and sequences all dependent on the key.

Keywords: Cryptography, Cryptographic Primitives, Cryptographic Sequences, Cryptographic Tables, Dynamic Substitution, Fast Encipherment, Cryptanalysis, Cryptanalytic Attacks, Cryptanalytic Complexity, Cryptanalytic Strength, Cryptanalytic Time, Cryptanalytic Work, Cryptanalytic Resistance, Cryptanalytic Security, Cryptanalytic Strength Estimation, Cryptanalytic Strength Measurement, Cryptanalytic Strength Estimation Methods, Cryptanalytic Strength Measurement Methods, Cryptanalytic Strength Estimation Techniques, Cryptanalytic Strength Measurement Techniques

Some results from Cellular Automata experiments are tabulated in Figure 1. Here there are two design variables: length (the number of elements), and width (the number of bits in each element). Note the wide variation in average cycle length, with no clear correlation to the design variables.

Figure 1. Exhaustive State Experiments: Cellular Automata

Len	Wid	Total States	Cycle States	Pct Cycles	Degen Cycles	Min Cycle	Avg Cycle	Avg Arc
4	1	16	11	68.8%	3	8	2.75	1.00
4	2	256	121	47.3%	9	8	5.26	1.00
4	3	4096	1331	32.5%	27	8	7.01	1.00
4	4	65536	14641	22.3%	81	8	7.70	1.00
5	2	1024	36	3.5%	1	5	4.50	9.15
5	3	32768	216	0.7%	1	5	4.91	21.5
6	2	4096	9	0.2%	9	-	1.00	583.9
7	2	16384	8464	51.7%	1	4	29.2	1.75
8	2	65536	2601	4.0%	9	8	30.6	24.4

The configuration with length 6 and width 2 presents an interesting case where virtually the entire state-space is located on arcs; the problem with this is that, within a finite state-space, any arc must eventually lead to a cycle. In this particular case, all the cycles are degenerate, and an arbitrary initialization may start out arbitrarily close to such a cycle. The *minimum* cycle length is our normal concern, for if an RNG is initialized arbitrarily, it may come up in a state within a minimum-length cycle, or anywhere on an arc leading to such a cycle. The minimum cycle length of CA RNG's seems rather short.

Some results of the x^2 mod N experiments are tabulated in Figures 2 and 3. The value of N is the only design variable. N is supposed to be the product of two large primes (*P* and *Q*); tiny primes were used to make exhaustive analysis practical. These systems are not permutation generators. The proportion of states in cycles seems fairly consistent: About 3 out of 4 of the total states are arc states, and each of these joins a cycle in just a single step. There are always 4 degenerate cycles. Because the x^2 mod N design might be configured for a particular cycle, maximum cycle length is also tabulated (instead of cycle percentage).

Figure 2.
Exhaustive State Experiments: Improperly Designed X^2 mod N

P	Q	Total States	Cycle States	Degen Cycles	Min Cycle	Avg Cycle	Max Cycle	Avg Arc
107	163	17441	4428	4	2	201.3	1404	1.00
107	211	22577	5724	4	2	73.4	156	1.00
127	131	16637	4224	4	2	11.4	12	1.00
127	139	17653	4480	4	2	35.0	66	1.00
139	167	23213	5880	4	2	245.0	902	1.00
139	211	29329	7420	4	2	50.1	132	1.00
151	199	30049	7600	4	2	30.4	60	1.00
151	211	31861	8056	4	2	24.0	60	1.00
163	191	31133	7872	4	2	49.2	108	1.00

For the *improper* x^2 mod N designs which are covered in Figure 2, *P* and *Q* are "Blum integers" [4; 96: 8], that is, they are primes congruent to 3 mod 4, but at most one is *special* as defined in Blum, Blum and Shub [21: 378]. Note the case of *P* = 127, *Q* = 131, which generates a sizable system with maximum cycle length of 12, and so is extraordinarily weak. In an improper design, a small increase in the value of one of the primes can result in a drastic *decrease* in strength.

Figure 3.
Exhaustive State Experiments: Properly Designed X^2 mod N

P	Q	Total States	Cycle States	Degen Cycles	Min Cycle	Avg Cycle	Max Cycle	Avg Arc
23	47	1081	288	4	10	24.0	110	1.00
23	167	3841	1008	4	10	100.8	410	1.00
23	359	8257	2160	4	10	216.0	890	1.00
23	719	16537	4320	4	10	360.0	1790	1.00
23	1439	33097	8640	4	10	720.0	3590	1.00
47	167	7849	2016	4	11	168.0	902	1.00
47	359	16873	4320	4	11	360.0	1958	1.00
47	719	33793	8640	4	11	540.0	1969	1.00
167	359	59953	15120	4	82	1512.0	7298	1.00

For the *proper* x^2 mod N designs covered in Figure 3, *P* and *Q* are "Blum integers" but they are also both *special* and developed as described in Blum, Blum and Shub. For a given system size they have longer minimum, maximum, and average cycle lengths than the improper designs of Figure 2. The minimum cycle length still seems rather short, but if we assume that the operational cycle is a configuration selection (instead of a key-related arbitrary setup), this may not be a problem. Alternately, there appears to be a formula for the period of the cycle including state 2, so if we use that particular cycle, it may be possible to increase the size of the system until the period is "long enough." The case of P = 47, Q = 719 is specifically prohibited in Blum, Blum and Shub [21: 378], yet gives no experimental indication of being "bad"; perhaps some other experiment would reveal a problem in this design.

Some results from the GFSR experiments are tabulated in Figure 4. Here there is an additional design variable, the selection of a primitive polynomial, although we are now constrained to use degrees which are Mersenne primes. Each experiment was performed using *every* primitive polynomial of the given degree, and the same results were obtained in each case.

Figure 4.
Exhaustive State Experiments: General Feedback Shift Register

Deg	Polys Tested	Wid	States	Pct Cycles	Degen Cycles	Min Cycle	Avg Cycle	Avg Arc
5	6	1	32	100.0%	1	31	16.0	0.00
5	6	2	1024	100.0%	1	31	30.1	0.00
5	6	3	32768	100.0%	1	31	30.97	0.00
7	18	1	128	100.0%	1	127	64.0	0.00
7	18	2	16384	100.0%	1	127	126.0	0.00

The degenerate cycle is the all-zeros case, which is prohibited; the rest of the states are part of a single cycle, and there are no arcs. Of course, each independent column *should* have at least one 1-bit, but cycle length is determined by any *one* non-zero column. Any such initialization yields the same cycle length, thus supporting *almost* arbitrary initialization (with respect to cycle length only, of course). The cycle length is set by the polynomial degree, and does not change as we widen the system. Consequently, if the width were to exceed the degree, only a subset of the possible output values could be produced. This would be an abnormal design, but it would seem that this worrisome concept must apply to some unknown extent in normal designs as well.

Some results from the Additive RNG experiments are tabulated in Figure 5. Again, each experiment was performed using *every* primitive polynomial of the given degree, and the same results were obtained in each case.

Figure 5.
Exhaustive State Experiments: Additive Generator

Deg	Polys Tested	Wid	States	Pct Cycles	Degen Cycles	Min Cycle	Avg Cycle	Avg Arc
5	6	1	32	96.9%	1	31	31.0	0.00
5	6	2	1024	96.9%	32	62	62.0	0.00
5	6	3	32768	96.9%	1024	124	124.0	0.00
7	18	1	128	99.2%	1	127	127.0	0.00
7	18	2	16384	99.2%	128	254	254.0	0.00

Here there is an initialization rule which eliminates degenerate cycles (a typically small part of the total). The remaining states form a number of isolated closed cycles with no arcs, again supporting almost-arbitrary initialization. The number of independent cycles varies with the width. However, since each cycle is equally long, this should be just as acceptable as a single long cycle. The cycle length also varies with the width, which is helpful.

7.3 Discussion (Prev|Next)

The Cellular Automata RNG contains cycles of various lengths, so its performance will vary, depending upon its initial state. Since cryptographic RNG's should ideally be initialized arbitrarily, CA RNG's may apparently come up in a weak short cycle, or on a short arc into a short cycle. Moreover, a larger CA RNG does not necessarily produce a stronger RNG; instead, particular special values for width and length may produce the strongest RNG. This evokes the realistic nightmare of a new search for the "best" such values after every new statistical analysis.

The x^2 mod N RNG is a special case: On the one hand, most of the comments on the CA RNG could also apply here; on the other, the extensive theory of operation seems to avoid the weakest designs, and should also allow the selection of a long operating cycle. Certainly an x^2 mod N design *must* follow the many intricate design requirements of Blum, Blum and Shub to avoid very significant weaknesses. Customization might consist of finding large special primes *P* and *Q*, or just the selection of a particular operating cycle within the context of a fixed value of *N*. (The number of usable states, here N/4, should be far larger than the total random numbers used by all fielded units over all installed time.)

The x^2 mod N generator seems to inherently include both short and degenerate cycles, so we cannot risk initializing x[0] arbitrarily. Thus, it is normally necessary to validate any x[0] "seed" to guarantee that it resides in a cycle which is "long enough." Alternately, user key initialization might select an arbitrary start state within a single configuration-selected cycle, if this can be done in real (user) time. Ultimately, such initialization would not be much different than that for a normal LFSR (since an arbitrary LFSR state is just a different start-state within the single maximal length cycle), but it is clearly *far* more complex than simply using arbitrary data as a convenient initialization state.

Both the GFSR and Additive RNG's were easy to design (even with primitive mod 2 polynomial selection), and were well behaved; there was no significant variation in performance with respect to initial state. Thus, an almost-arbitrary initialization will always produce similar performance. In these designs, a longer polynomial always produced longer cycles; in the Additive RNG, a wider register also always produced longer cycles. In both designs, all of the states were in cycles, and there were no dangerous short cycles. There were also no arcs, and thus no arcs into short or degenerate cycles. Both of these designs can easily be customized through the use of a new primitive polynomial, and the customized design will carry the same performance guarantees.

Certainly, the CA and x^2 mod N designs would have to be considered more secure *as they are normally proposed*; both of these designs are "Class C" RNG's in that the designer is urged to expose only a portion of their state change on every step. But the Additive RNG could be similarly restricted (especially since it is so easily expanded), and both the GFSR and Additive designs are clearly usable with any other form of isolation mechanism.

8. SUMMARY (Prev|Next)

A great deal of work remains to be done to establish the cryptographic performance of the various random number generators.

Currently, the Additive RNG seems attractive since it is easily initialized from a user key, easily expanded to huge amounts of internal state, easily customized, reasonably efficient, and has a guaranteed minimum cycle length. The Additive RNG will remain attractive precisely until a much better mechanism is found or a fatal flaw discovered in this one.

The x^2 mod N generator may also be expanded and customized, but seems to require far more computational effort at every stage of the process.

Other RNG techniques could also be attractive, if a good theoretical basis for their minimum cycle lengths could be found, and the design made "long enough." Implementation difficulty and computational efficiency could then be compared to make a reasonable choice.

Because most RNG's can be cryptanalyzed, steps must usually be taken to prevent penetration, often through the use of some sort of isolation mechanism. Here again, much more theoretical work is needed on the ability of various mechanisms to protect an RNG from analysis.

9. COMMENTS (Prev|Next)

First, much of the modern theoretical work seems oriented at solving "the problem of cryptography" in a single step; this work often implies systems which are slow and complex. One alternative is to design individual mechanisms or processes which solve one aspect of a cryptographic system to some known extent; then multiple such devices might be arranged to completely solve the whole problem. Such a design might be fast, and also better partitioned to support a deep understanding of the overall design. This approach should also be more amenable to deep mathematical analysis than trying to solve the whole problem at once.

Next, both mathematicians and engineers seek simple systems. However, as we have seen, a simple *formula* in no way implies a simple *system*: Clearly, the complexity of the lexical mathematical representation is not isomorphic to the complexity of the realization. Systems based on apparently trivial equations can place surprisingly massive requirements on an actual implementation. Even simple integer arithmetic represents a fairly complex system at the binary gate level [e.g., 2: 23]; perhaps systems based on integer arithmetic are inherently complex (even the successful period analysis of the simple Additive RNG seems rather difficult [114]). As a goal, mathematicians might strive to work with lexical representations which are more closely related to the complexity of the resulting system. The ultimate system must be relatively simple if we are to avoid dangerous hidden problems; consequently, even non-mathematicians should be able to understand those systems.

Finally, almost any encryption or random number implementation has some mathematical claim, if only a hand-waving "there are a large number of possibilities, so this must be secure." More formal mathematical models may indeed imply strength, but *only* if the resulting mechanism implements the mathematical model completely and exactly. Unfortunately, it is easy to make an implementation mistake in a complex design: Some errors of understanding do withstand review, some errors of design do result in "working" systems, and even massive testing can miss various types of problems. Both correct and incorrect implementations can look and perform remarkably alike, differing only in small but crucial details.

It would certainly be a benefit for system builders if the mathematicians, as part of the original work, would define a set of tests which would be sufficient to completely validate the implementation of their mathematical model. Because of the deep mathematical background often assumed in the original work, it is not always possible for a non-mathematician to do this correctly.

What would the mathematicians get out of such validation? Well, the goal would be to establish a correlation between a theoretical and a real system; the real system could then be used for experiments, in ways similar to non-linear dynamics experiments. Small systems could be exhaustively tested for conformance to predicted results; results could provide direct insight into the math. Such experiments might reveal entire new classes of relationship, perhaps some that could otherwise be found only with great difficulty. There can be serious advantages to having a validated realization to play with.

10. REFERENCES (Prev|Next)

1. Agnew, G. 1986. Random Sources for Cryptographic Systems. *Advances in Cryptology: CRYPTO '85 Proceedings.* 77-81. Springer-Verlag: Berlin / New York.

2. Aho, A., J. Hopcroft and J. Ullman. 1974. *The Design and Analysis of Computer Algorithms.* Addison-Wesley: Reading, Massachusetts.

3. Akl, S. and H. Meijer. 1984. A Fast Pseudo Random Permutation Generator. *Advances in Cryptology: CRYPTO '84 Proceedings.* 269-275. Springer-Verlag: Berlin / New York.

4. Alexi, W., B. Chor, O. Goldreich, and C. Schnorr. 1984. RSA/Rabin Bits are 1/2 + 1/poly(log N) Secure (Extended Abstract). *25th IEEE Symposium on the Foundations of Computer Science.* 449-457.

5. Allen, T. 1983. Controlled, but Rational, Phase-Locking Responses of a Simple Time-Base Oscillator. *IEEE Transactions on Circuits and Systems.* CAS-30: 627-632.

6. Allender, E. 1987. Some Consequences of the Existence of Pseudorandom Generators. *Proceedings of the Nineteenth Annual ACM Symposium on Theory of Computing.* 151-159.

7. Anderson, R. 1990. Solving a Class of Stream Ciphers. *Cryptologia.* 14: 285-288.

8. Arazi, B. 1988. *A Commonsense Approach to the Theory of Error Correcting Codes.* MIT Press: Cambridge, Mass.

9. *Artificial Life.* 1989. Editor: C. Langton. Addison-Wesley: New York.

10. Arvillias, A. and D. Maritsas. 1978. Partitioning the Period of a Class of *m*-Sequences and Application to Pseudorandom Number Generation. *Journal of the Association for Computing Machinery.* 25: 675-686.

11. Bays, C. and S. Durham. 1976. Improving a Poor Random Number Generator. *ACM Transactions on Mathematical Software.* 2: 59-64.

12. Beker, H. and F. Piper. 1982. *Cipher Systems.* John Wiley: New York.

13. Beker, K. and M. Dorfler. 1989. *Dynamic systems and fractals.* Cambridge University Press, New York.

14. Bell, E. 1951. *Mathematics, Queen and Servant of Science.* Microsoft Press: Redmond, Washington.

15. Bell, T. 1986. Better OPM/L Text Compression. *IEEE Transactions on Communications.* COM-34: 1176-1182.

16. Ben-Or, M. 1981. Probabilistic algorithms in finite fields. *Proceedings of the 22nd IEEE Foundations of Computer Science Symposium.* 394-398.

17. Berlekamp, E. 1984. *Algebraic Coding Theory.* Aegean Park Press: Laguna Hills, CA.

18. Berlinski, D. 1988. *Black Mischief: Language, Life, Logic, Luck.* 2nd Edition. Harcourt Brace Jovanovich: Boston / New York.

19. Blahut, R. 1983. *Theory and Practice of Error Control Coding.* Addison-Wesley: Reading, Mass.

20. Blum, L., M. Blum and M. Shub. 1983. Comparison of Two Pseudo-Random Number Generators. *Advances in Cryptology: CRYPTO '82 Proceedings.* Plenum Press: New York. 61-78.

21. Blum, L., M. Blum and M. Shub. 1986. A Simple Unpredictable Pseudo-Random Number Generator. *SIAM Journal on Computing.* 15: 364-383.

22. Blum, M. 1982. Coin flipping by telephone. *Proceedings of the 24th IEEE Spring COMPCON.* 133-137.

23. Blum, M. 1984. Independent unbiased coin flips from a correlated biased source: a finite state markov chain. *25th IEEE Symposium on the Foundations of Computer Science.* 425-433.

24. Blum, M. and S. Micali. 1984. How to Generate Cryptographically Strong Sequences of Pseudo-Random Bits. *SIAM Journal on Computing.* 13: 850-864.

25. Boyar, J. 1989. Inferring Sequences Produced by Pseudo- Random Number Generators. *Journal of the ACM.* 36: 129-141.

26. Bright, H. and R. Enison. 1979. Quasi-Random Number Sequences from a Long-Period TLP Generator with Remarks on Application to Cryptography. *ACM Computing Surveys.* 11: 357-370.

27. Burks, A. 1970. *Essays on Cellular Automata.* University of Illinois Press: Urbana / Chicago / London.

28. Calmet, J. and R. Loos. 1980. An Improvement of Rabin's Probabilistic Algorithm for Generating Irreducible Polynomials over GF(p). *Information Processing Letters.* 11: 94-95.

29. Campbell, J. 1982. *Grammatical Man: Information, Entropy, Language and Life.* Simon and Schuster: New York.

30. Chaitin, G. 1975. Randomness and Mathematical Proof. *Scientific American.* 232(5): 47-52.

31. Chaitin, G. 1987. *Algorithmic Information Theory.* Cambridge University Press: Cambridge / New York.

32. Chor, B., O. Goldreich and S. Goldwasser. 1985. The Bit Security of Modular Squaring Given Partial Factorization of the Modulos. *Advances in Cryptology: CRYPTO '85 Proceedings.* 448-457. Springer-Verlag: Berlin / New York.

33. Chor, B. and O. Goldreich. 1988. Unbiased Bits from Sources of Weak Randomness and Probabilistic Communication Complexity. *SIAM Journal on Computing.* 17: 230-261.

34. Ciarcia, S. 1986. Build a Hardware Data Encryptor. *Byte.* September. 97-111.

35. Cleary, J. and I. Witten. 1984. Data Compression Using Adaptive Coding and Partial String Matching. *IEEE Transactions on Communications.* COM-32: 396-402.

36. Collings, B. and G. Hembree. 1986. Initializing Generalized Feedback Shift Register Pseudorandom Number Generators. *Journal of the ACM.* 33: 707-711, 34: 1001.

37. Collings, B. 1987. Compound Random Number Generators. *Journal of the American Statistical Association.* 82 (398): 525-527.

38. Compagner, A. and A. Hoogland. 1987. Maximum-Length Sequences, Cellular Automata, and Random Numbers. *Journal of Computational Physics.* 71: 391-428.

39. Cormack, G. and R. Horspool. 1984. Algorithms for Adaptive Huffman Codes. *Information Processing Letters.* 18: 159-165.

40. Costas, J. 1979. Cryptography in the Field, Part 2: Using the Pocket Calculator. *Byte.* March 1979. 157.

41. *Cryptology Yesterday, Today, and Tomorrow.* 1987. Editors: C. Deavours, D. Kahn, L. Kruh, G. Mellen and B. Winkle. Artech House: Norwood, Mass.

42. Davies, D. and W. Pierce. 1984. *Security for Computer Networks.* John Wiley: New York.

43. Davis, W. 1966. Automatic Delay Changing Facility for Delayed *m*-Sequences. *Proceedings of the IEEE.* 54: 913-914.

44. Deavours, C. 1977. Unicity Points in Cryptanalysis. *Cryptologia.* 1(1). (Also in [41: 359-378].)

45. Denning, D. 1982. *Cryptography and Data Security.* Reading, Mass: Addison-Wesley.

46. de Visme, G. 1971. *Binary Sequences.* The English Universities Press: London.

47. Diaconis, P. and R. Grahm. 1977. Spearman's Footrule as a Measure of Disarray. *Journal of the Royal Statistical Society.* Series B. 39: 262-268

48. Diaconis, P. and M. Shahshahavi. 1981. Generating a Random Permutation with Random Transpositions. *Z. Wahscheinlichkeitstheorie.* 57: 159-179. (Springer-Verlag.)

49. Diffie, W. and M. Hellman. 1976. New Directions in Cryptography. *IEEE Transactions on Information Theory.* IT22: 644-654.

50. Durstenfeld, R. 1964. Algorithm 235, Random Permutation, Procedure SHUFFLE. *Communications of the ACM.* 7: 420.

51. Erber, T., P. Everett and P. Johnson. 1979. The Simulation of Random Processes on Digital Computers with Cebysev Mixing Transformations. *Journal of Computational Physics.* 32: 168-211.

52. Erber, T., T. Rynne, W. Darsow and M. Frank. 1983. The Simulation of Random Processes on Digital Computers: Unavoidable Order. *Journal of Computational Physics.* 49: 349-419.

53. Erber, T. and S. Putterman. 1985. Randomness in quantum mechanics -- nature's ultimate cryptogram? *Nature.* 318: 41-43.

54. Etzion, T. and A. Lempel. 1984. Algorithms for the Generation of Full-Length Shift-Register Sequences. *IEEE Transactions on Information Theory.* IT-30: 480-484.

55. Feldman, F. 1988. Fast Spectral Tests for Measuring Nonrandomness and the DES. *Advances in Cryptology: CRYPTO '87 Proceedings.* 243-254. Springer-Verlag: Berlin / New York.

56. Fischer, E. 1981. A Theoretical Measure of Cryptographic Performance. *Cryptologia.* 5(1). (Also in [41: 421-426].)

57. Fischer, E. 1981. Measuring Cryptographic Performance with Production Processes. *Cryptologia.* 5(3). (Also in [41: 421-426].)

58. Fushimi, M. and S. Tezuka. 1983. The *k*-Distribution of Generalized Feedback Shift Register Pseudorandom Numbers. *Communications of the ACM.* 26(7): 516-523.

59. Fushimi, M. 1983. Increasing the Orders of Equidistribution of the Leading Bits of the Tausworthe Sequence. *Information Processing Letters.* 16: 189-192.

60. Fushimi, M. 1988. Designing a Uniform Random Number Generator Whose Subsequences are *k*-Distributed. *SIAM Journal on Computing.* 17: 1988.

61. Fushimi, M. 1989. An Equivalence Relation between Tausworthe and GFSR Sequences and Applications. *Applied Mathematics Letters.* 2(2): 135-137.

62. Gallager, R. 1978. Variations on a Theme by Huffman. *IEEE Transactions on Information Theory.* IT-24: 668-674.

63. Gardner, M. 1983. *Wheels, Life and Other Mathematical Amusements.* W. H. Freeman: New York.

64. Geffe, P. 1973. How to protect data with ciphers that are really hard to break. *Electronics.* 46(1): 99-101.

65. Gleick, J. 1987. *Chaos: Making a New Science.* Viking Penguin: New York.

66. Goldreich, O., S. Goldwasser and S. Micali. 1984. How to Construct Random Functions (Extended Abstract). *25th IEEE Symposium on the Foundations of Computer Science.* 464-479.

67. Goldreich, O., H. Krawczyk and M. Luby. 1988. On the Existence of Pseudorandom Generators (Extended Abstract). *29th IEEE Symposium on the Foundations of Computer Science.* 12-24.

68. Goldwasser, S. and S. Micali. 1984. Probabilistic Encryption. *Journal of Computer and System Sciences.* 28: 270-299.

69. Goldwasser, S. and J. Kilian. 1986. Almost All Primes Can be Quickly Certified. *18th Symposium on the Theory of Computing.* 316-329.

70. Gollmann, D. and W. Chambers. 1989. Clock-Controlled Shift Registers: A Review. *IEEE Journal on Selected Areas in Communications.* 7: 525-533.

71. Golomb, S. 1982 (original publication 1967). *Shift Register Sequences,* Revised Edition. Aegean Park Press: Laguna Hills, CA.

72. Groth, E. 1971. Generation of Binary Sequences With Controllable Complexity. *IEEE Transactions on Information Theory.* IT-17: 288-296.

73. Guinier, D. 1989. A Fast Uniform "Astronomical" Random Number Generator. *SIGSAC Review* (ACM Special Interest Group on Security Audit & Control). 7(1): 1-13. ACM Press: New York.

74. Günther, C. 1987. Alternating Step Generators Controlled by De Bruijn Sequences. *Advances in Cryptology: EUROCRYPT '87 Proceedings.* 5-14. Springer-Verlag: Berlin / New York.

75. Hastad, J. and A. Shamir. 1985. The Cryptographic Security of Truncated Linearly Related Variables. *17th ACM Symposium on Theory of Computing.* 356-362.

76. Hellman, M. 1977. An Extension of the Shannon Theory Approach to Cryptography. *IEEE Transactions on Information Theory.* IT-23: 289-294.

77. Hellman, M. 1982. Cryptographic Key Size Issues. *Proceedings of the (24th) IEEE Spring COMPCON 1982.* 130-137.

78. Hemmati, F. 1982. A Large Class of Nonlinear Shift Register Sequences. *IEEE Transactions on Information Theory.* IT-28: 355-359.

79. Herlestam, T. 1983. On Using Prime Polynomials in Crypto Generators. *Cryptography. Proceedings, Burg Feuerstein 1982.* Springer-Verlag: Berlin / New York.

80. Hofstadter, D. 1979. *Gödel, Escher, Bach: an Eternal Golden Braid.* Basic Books: New York.

81. Horowitz, P. and W. Hill. 1989. *The Art of Electronics,* 2nd Edition. Cambridge University Press: New York.

82. Hosack, J. 1986. The Use of Cebysev Mixing to Generate Pseudo-random Numbers. *Journal of Computational Physics.* 67: 482-486.

83. Huffman, D. 1952. A Method for the Construction of Minimum-Redundency Codes. *Proceedings of the IRE.* 40: 1098-1101.

84. Jay, F., ed. 1977. *IEEE Dictionary of Electrical and Electronics Terms,* 2nd Ed. IEEE / John Wiley: New York.

85. Johnsen, V. and K. Kjeldsen. 1973. Loop-free Compositions of Certain Finite Automata. *Information and Control.* 22: 303-319.

86. Jueneman, R. 1987. A High Speed Manipulation Detection Code. *Advances in Cryptology: CRYPTO '86 Proceedings.* 327-346. Springer-Verlag: Berlin / New York.

87. Jueneman, R. 1987. Electronic Document Authentication. *IEEE Network Magazine.* 1(2): 17-23.

88. Kahn, D. 1967. *The Codebreakers.* Macmillan: New York.

89. Key, E. 1976. An Analysis of the Structure and Complexity of Nonlinear Binary Sequence Generators. *IEEE Transactions on Information Theory.* IT-22: 732-736.

90. Kirkpatrick, S. and E. Stoll. 1981. Note: A Very Fast Shift-Register Sequence Random Number Generator. *Journal of Computational Physics.* 40: 517-526.

91. Knuth, D. 1981. *The Art of Computer Programming,* Vol. 2, *Seminumerical Algorithms.* 2nd ed. Addison-Wesley: Reading, Massachusetts.

92. Knuth, D. 1985. Deciphering a Linear Congruential Encryption. *IEEE Transactions on Information Theory.* IT-31: 49-52.

93. Kolmogorov, A. 1965. Three approaches to the quantitative definition of information. *Problems of Information Transmission.* 1: 1-7.

94. Koopman, R. 1986. The Orders of Equidistribution of Subsequences of Some Asymptotically Random Sequences. *Communications of the ACM.* 29: 802-806.

95. Kranakis, E. 1986. *Primality and Cryptography.* John Wiley: New York.

96. Landau, S. 1988. Zero Knowledge and the Department of Defense. *Notices of the American Mathematical Society.* 35: 5-12.

97. Langdon, G. 1983. A Note on the Ziv-Lempel Model for Compressing Individual Sequences. *IEEE Transactions on Information Theory.* IT-29: 284-286.

98. L'Ecuyer, P. 1988. Efficient and Portable Combined Random Number Generators. *Communications of the ACM.* 31(6): 742-749, 774.

99. L'Ecuyer, P. and R. Proulx. 1989. About Polynomial-Time "Unpredictable" Generators. *Proceedings of the 1989 Winter Simulation Conference.* 467-476. IEEE Press: New York.

100. L'Ecuyer, P. 1990. Random numbers for simulation. *Communications of the ACM.* 33(10): 86-97.

101. Letham, L., D. Hoff and A. Folmsbee. 1986. A 128k EPROM Using Encryption of Pseudorandom Numbers to Enable Read Access. *IEEE Journal of Solid-State Circuits.* SC-21: 881-888.

102. Lempel, A., and J. Ziv. 1976. On the Complexity of Finite Sequences. *IEEE Transactions on Information Theory.* IT-22: 75-81.

103. Levin, L. 1985. One-Way Functions and Pseudorandom Generators. *17th ACM Symposium on Theory of Computing.* 363-365.

104. Levine, J. and V. Brawley. 1977. Some Cryptographic Applications of Permutation Polynomials. *Cryptologia.* 1: 76-92.

105. Levine, J. and R. Chandler. 1987. Some Further Cryptographic Applications of Permutation Polynomials. *Cryptologia.* 11: 211-217.

106. Lewis, T. and W. Payne. 1973. Generalized Feedback Shift Register Pseudorandom Number Algorithm. *Journal of the ACM.* 20: 456-468.

107. Long, D. and A. Wigderson. 1983. How Discreet is the Discrete Log? *15th ACM Symposium on Theory of Computing.* 413-420.

108. Lu, S. 1979. The Existence of Good Cryptosystems for Key Rates Greater than Message Redundancy. *IEEE Transactions on Information Theory.* IT-25: 475-477.

109. MacLaren, M. and G. Marsaglia. 1965. Uniform Random Number Generators. *Journal of the Association for Computing Machinery.* 12: 83-89.

110. MacWilliams, F. and N. Sloane. 1976. Pseudo-Random Sequences and Arrays. *Proceedings of the IEEE.* 64: 1715-1729.

111. MacWilliams, F. and N. Sloane. 1977. *The Theory of Error-Correcting Codes.* North Holland: Amsterdam / New York.

112. Marsaglia, G., and T. Bray. 1968. One-Line Random Number Generators and Their Use in Combinations. *Communications of the ACM.* 11: 757-759.

113. Marsaglia, G. 1984. A current view of random number generators. *Proceedings of the Sixteenth Symposium on the Interface Between Computer Science and Statistics.* 3-10.

114. Marsaglia, G. and L. Tsay. 1985. Matrices and the Structure of Random Number Sequences. *Linear Algebra and its Applications.* 67: 147-156.

115. Marsaglia, G. and A. Zaman. 1990. Toward a Universal Random Number Generator. *Statistics and Probability Letters.* 8: 35-39.

116. Martin-L”f, P. 1966. The Definition of Random Sequences. *Information and Control.* 9: 602-619.

117. Massey, J. 1969. Shift-Register Synthesis and BCH Decoding. *IEEE Transactions on Information Theory.* IT-15: 122-127.

118. Massey, J. 1988. An Introduction to Contemporary Cryptology. *Proceedings of the IEEE.* 76: 533-549.

119. Matthews, R. 1989. On the Derivation of a "Chaotic" Encryption Algorithm. *Cryptologia.* 13: 29-42.

120. McCoy, N. 1972. *Fundamentals of Abstract Algebra.* Allyn and Bacon: Boston.

121. Meier, W. and O. Staffelbach. 1988. Fast Correlation Attacks on Stream Ciphers (extended abstract). *Advances in Cryptology: EUROCRYPT '88 Proceedings.* 301-314. Springer-Verlag: Berlin / New York.

122. Mellen, G. 1973. Cryptology, Computers, and Common Sense. *Proceedings of the National Computer Conference.* 1973: 569-579.

123. Meyer, C. and W. Touchman. 1972. Pseudorandom codes can be cracked. *Electronic Design 23.* (Nov): 74-76.

124. Meyer, C. and S. Matyas. 1982. *Cryptography: A New Dimension in Computer Data Security.* John Wiley: New York.

125. Mitchell, D. 1990. Nonlinear Key Generators. *Cryptologia.* 14: 350-354.

126. Mund, S., D. Gollmann, and T. Beth. 1987. Some Remarks on the Cross Correlation Analysis of Pseudo Random Generators. *Advances in Cryptology: EUROCRYPT '87 Proceedings.* 25-35. Springer-Verlag: Berlin / New York.

127. Nichols, E., J. Nichols, and K. Musson. 1982. *Data Communications for Microcomputers.* McGraw-Hill: New York.

128. Nicolis, G. and I. Prigogine. 1989. *Exploring Complexity.* W. H. Freeman and Company: New York.

129. Niederreiter, H. 1987. A Statistical Analysis of Generalized Feedback Shift Register Pseudorandom Number Generators. *SIAM Journal on Scientific and Statistical Computing.* 8: 1035-1051.

130. Nisan, N. and A. Wigderson. 1988. Hardness vs. Randomness (Extended Abstract). *29th IEEE Symposium on Foundations of Computer Science.* 2-11.

131. Page, E. 1967. A note on generating random permutations. *Applied Statistics.* 16: 273-274.

132. Park, S. and K. Miller. 1988. Random Number Generators: Good Ones Are Hard To Find. *Communications of the ACM.* 31: 1192-1201.

133. Patterson, W. 1987. *Mathematical Cryptology.* Rowan & Littlefield: Totowa, N.J.

134. Pearson, P. 1988. Cryptanalysis of the Ciarcia Circuit Cellar Data Encryptor. *Cryptologia.* 12: 1-9.

135. Pearson, P. 1990. Fast Hashing of Variable-Length Text Strings. *Communications of the ACM.* 33: 677-680.

136. Pickover, C. 1988. Pattern Formation and Chaos in Networks. *Communications of the ACM.* 31: 136-151.

137. Pierce, J. 1980. *An Introduction to Information Theory.* 2nd Ed. Dover Publications: New York.

138. PKZIP110.EXE, *APPNOTE.TXT* (shareware file compression utility documentation). 1989. PKWARE: Glendale, WI.

139. Plackett, R. 1968. Random Permutations. *Journal of the Royal Statistical Society.* Series B. 30: 517-534.

140. Pless, V. 1977. Encryption Schemes for Computer Confidentiality. *IEEE Transactions on Computers.* C-26: 1133-1136.

141. Poundstone, W. 1985. *The Recursive Universe.* Contemporary Books: Chicago.

142. Rabin, M. 1980. Probabilistic Algorithms in Finite Fields. *SIAM Journal on Computing.* 9: 273-280.

143. Ramabadran, T. and S. Gaitonde. A Tutorial on CRC Computations. *IEEE Micro.* August: 62-75.

144. Rao, C. 1961. Generation of Random Permutations of Given Number of Elements Using Random Sampling Numbers. *Sankhya.* Series A. 23: 305-307.

145. Rasband, S. 1990. *Chaotic Dynamics of Nonlinear Systems.* John Wiley & Sons: New York.

146. Reed, I. and R. Turn. 1969. A Generalization of Shift-Register Sequence Generators. *Journal of the Association for Computing Machinery.* 16: 461-473.

147. Reeds, J. 1977. "Cracking" a Random Number Generator. *Cryptologia.* 1(1). (Also [41: 509-515].)

148. Reif, J. and D. Tygar. 1988. Efficient Parallel Pseudorandom Number Generation. *SIAM Journal on Computing.* 17: 404-411.

149. Retter, C. 1984. Cryptanalysis of a MacLaren-Marsaglia System. *Cryptologia.* 8:97-108. Also see letters, 8: 374-378.

150. Retter, C. 1985. A Key Search Attack on MacLaren-Marsaglia Systems. *Cryptologia.* 9: 114-130.

151. Rietman, E. 1989. *Exploring the Geometry of Nature.* Windcrest Books, Blue Ridge Summit, PA.

152. Ritter, T. 1990. Substitution Cipher with Pseudo-Random Shuffling: The Dynamic Substitution Combiner. *Cryptologia.* 14(4): 289-303.

153. Ritter, T. 1991. Transposition Cipher with Pseudo-Random Shuffling: The Dynamic Transposition Combiner. *Cryptologia.* 15(1):1-17.

154. Rivest, R, A. Shamir, and L. Adleman. 1978. A Method for Obtaining Digital Signatures and Public Key Cryptosystems. *Communications of the ACM.* 21: 120-126.

155. Rivest, R. 1980. "Forwards and Backwards" Encryption. *Cryptologia.* 4(1). (Also [41: 433-437].)

156. Robbins, D. and E. Bolker. 1981. The bias of three pseudo- random shuffles. *Aequationes Mathematicae.* 22: 268-292.

157. Rubin, F. 1978. Computer Methods for Decrypting Random Stream Ciphers. *Cryptologia.* 2(3). (Also [41: 493-508].)

158. Rubin, F. 1979. Decrypting a Stream Cipher Based on J-K Flip-Flops. *IEEE Transactions on Computers.* C28: 483-487. (Also *Cryptologia* 5(1), and [41: 283-293].)

159. Rudich, S. 1985. Inferring the structure of a Markov Chain from its output. *26th IEEE Symposium on the Foundations of Computer Science.* 321-326.

160. Rueppel, R. 1985. Correlation Immunity and the Summation Generator. *Advances in Cryptology: CRYPTO '85 Proceedings.* 260-272. Springer-Verlag: Berlin / New York.

161. Rueppel, R. 1987. Linear Complexity and Random Sequences. *Advances in Cryptology: CRYPTO '86 Proceedings.* 167-188. Springer-Verlag: Berlin / New York.

162. Rueppel, R. and O. Staffelbach. 1987. Products of Linear Sequences with Maximal Complexity. *IEEE Transactions on Information Theory.* IT-33: 124-131.

163. Rueppel, R. 1988. When Shift Registers Clock Themselves. *Advances in Cryptology: EUROCRYPT '87 Proceedings.* 53-64. Springer-Verlag: Berlin / New York.

164. Sandelius, M. 1962. A simple randomization procedure. *Journal of the Royal Statistical Society.* Series B. 24: 472-481.

165. Santha, M. and U. Vazirani. 1986. Generating Quasi-random Sequences from Semi-random Sources. *Journal of Computer and System Sciences.* 33: 75-87.

166. Sastri, K. 1975. Specified sequence linear feedback shift registers. *International Journal of Systems Science.* 6: 1009-1019.

167. Shamir, A. 1981. On the generation of cryptographically strong pseudo-random sequences. *8th International Colloquium on Automata, Language, and Programming.* 544-550.

168. Shannon, C. and W. Weaver. 1949. *The Mathematical Theory of Communication.* University of Illinois Press: Urbana, Chicago.

169. Shannon, C. 1949. Communication Theory of Secrecy Systems. *Bell System Technical Journal.* 28: 656-715.

170. Siegenthaler, T. 1984. Correlation-Immunity of Nonlinear Combining Functions for Cryptographic Applications. *IEEE Transactions on Information Theory.* IT-30: 776-780.

171. Siegenthaler, T. 1985. Decrypting a Class of Stream Ciphers Using Ciphertext Only. *IEEE Transactions on Computers.* C34: 81-85.

172. Siegenthaler, T. 1986. Cryptanalysts Representation of Nonlinearly Filtered ML-Sequences. *Advances in Cryptology: EUROCRYPT '85 Proceedings.* 103-110. Springer-Verlag: Berlin / New York.

173. Siegenthaler, T. 1986. Design of Combiners to Prevent Divide and Conquer Attacks. *Advances in Cryptology: CRYPTO '85 Proceedings.* Springer-Verlag: Berlin / New York.

174. Siegenthaler, T., R. Forre, and A. Kleiner. 1987. Generation of Binary Sequences with Controllable Complexity and Ideal *r*-Tupel Distribution. *Advances in Cryptology: EUROCRYPT '87 Proceedings.* 15-23. Springer-Verlag: Berlin / New York.

175. Simmons, G. 1979. Symmetric and Asymmetric Encryption. *Computing Surveys.* 11: 305-330.

176. Simmons, G. 1988. A Survey of Information Authentication. *Proceedings of the IEEE.* 76: 603-620.

177. Sloane, N. 1983. Encrypting by Random Rotations. *Cryptography. Proceedings, Burg Feuerstein 1982.* 71-128. Springer-Verlag: Berlin / New York.

178. Stahnke, W. 1973. Primitive Binary Polynomials. *Mathematics of Computation.* 27: 977-980.

179. Stern, J. 1987. Secret Linear Congruential Generators are Not Cryptographically Secure. *28th Annual Symposium on Foundations of Computer Science.* 421-426.

180. Storer, J. and T. Szymanski. 1982. Data Compression via Textual Substitution. *Journal of the ACM.* 29: 928-951.

181. Swift, J. 1960. Construction of Galois Fields of Characteristic Two and Irreducible Polynomials. *Mathematics of Computation.* 14: 99-103.

182. Tanenbaum, A. 1981. *Computer Networks.* Prentice-Hall: Englewood Cliffs, NJ.

183. Tang, Y., A. Mees and L. Chua. 1983. Synchronization and Chaos. *IEEE Transactions on Circuits and Systems.* CAS-30: 620-626.

184. Tausworthe, R. 1965. Random Numbers Generated by Linear Recurrence Modulo Two. *Mathematics of Computation.* 19: 201-209.

185. Tezuka, S. 1987. Walsh-Spectral Test for GFSR Pseudorandom Numbers. *Communications of the ACM.* 30: 731-735.

186. Tezuka, S. 1987. On the Discrepancy of GFSR Pseudorandom Numbers. *Journal of the Association for Computing Machinery.* 34: 939-949.

187. Tezuka, S. 1988. On Optimal GFSR Pseudorandom Number Generators. *Mathematics of Computation.* 50: 531-533.

188. Tootill, J., W. Robinson, and A. Adams. 1971. The Runs Up-and-Down Performance of Tausworthe Pseudo-Random Number Generators. *Journal of the Association for Computing Machinery.* 18:381-399.

189. Tootill, J., W. Robinson, and D. Eagle. 1973. An Asymptotically Random Tausworthe Sequence. *Journal of the ACM.* 20: 469-481.

190. Vahle, M. and L. Tolendino. 1982. Breaking a Pseudo Random Number Based Cryptographic Algorithm. *Cryptologia.* 6: 319-328.

191. Vazirani, U. and V. Vazirani. 1983. Trapdoor Pseudo-random Number Generators with Applications to Protocol Design. *24th IEEE Symposium on the Foundations of Computer Science.* 23-30.

192. Vazirani, U. and V. Vazirani. 1985. Efficient and Secure Pseudo-Random Number Generation. (extended abstract) *Advances in Applied Cryptology: Proceedings of CRYPTO 84.* 193-202. Springer-Verlag: Berlin / New York.

193. Vazirani, U. 1985. Towards a Strong Communication Complexity Theory or Generating Quasi-Random Sequences from Two Communicating Slightly-Random Sources. *Proceedings of the 17th Annual ACM Symposium on the Theory of Computing.* 366-378.

194. Vernam, G. 1926. Cipher Printing Telegraph Systems. *Transactions of the AIEE.* 45: 295-301.

195. von Neumann, J. 1963. Various techniques used in connection with random digits. *John von Neumann, Collected Works.* 5: 768-770. A. H. Taub, Ed. MacMillan: New York.

196. Wagner, P., P. Putter and M. Cain. 1987. Large-Scale Randomization Techniques. *Advances in Cryptology: CRYPTO '86 Proceedings.* 394-404. Springer-Verlag: Berlin / New York.

197. Watson, E. 1962. Primitive Polynomials (Mod 2). *Mathematics of Computation.* 16: 368-369.

198. Wayner, P. 1987. Building an Encryption System. *Computer Language.* December: 67-72.

199. Welch, T. 1984. A Technique for High-Performance Data Compression. *Computer.* June, 1984: 8-19.

200. Westlake, W. 1967. A Uniform Random Number Generator Based on the Combination of Two Congruential Generators. *Journal of the Association for Computing Machinery.* 14: 337-340.

201. Wheeler, D. 1989. Problems with Chaotic Cryptosystems. *Cryptologia.* 13: 243-250.

202. Wichmann, B. and I. Hill. 1982. An Efficient and Portable Pseudo-Random Number Generator. Applied *Statistics.* 33: 706-711. (Also see comments and responses 33: 123, 34: 198-200 and 35: 89.)

203. Wichmann, B. and D. Hill. 1987. Building a Random-Number Generator. *Byte.* March, 1987. 127-128.

204. Wikramaratna, R. 1989. ACORN -- A New Method for Generating Sequences of Uniformly Distributed Pseudo-random Numbers. *Journal of Computational Physics.* 83: 16-31.

205. Winternitz, R. 1984. A Secure One-Way Hash Function Built from DES. *1984 IEEE Symposium on Secrecy and Privacy.* 88-90.

206. Wolfram, S. 1985. Cryptography with Cellular Automata (extended abstract). *Advances in Cryptology: CRYPTO '85 Proceedings.* Springer-Verlag, 1986. 429-432.

207. Wolfram, S. 1986. Random Sequence Generation by Cellular Automata. *Advances in Applied Mathematics.* 7: 123-169.

208. Yao, A. 1982. Theory and Applications of Trapdoor Functions. *Proceedings of the 23rd IEEE Symposium on Foundations of Computer Science.* 80-91.

209. Yuen, C. 1977. Testing Random Number Generators by Walsh Transform. *IEEE Transactions on Computers.* C-26: 329-333.

210. Zeisel, H. 1986. A Remark on Algorithm AS 183. An Efficient and Portable Pseudo-random Number Generator. *Applied Statistics.* 35: 89.

211. Zierler, N. and J. Brillhart. 1968. On Primitive Trinomials (Mod 2). *Information and Control.* 13: 541-554.

212. Zierler, N. and J. Brillhart. 1969. On Primitive Trinomials (Mod 2), II. *Information and Control.* 14: 566-569.

213. Ziv, J. and A. Lempel. 1977. A Universal Algorithm for Sequential Data Compression. *IEEE Transactions on Information Theory.* IT-23: 337-343.

BIOGRAPHICAL SKETCH (Prev)

Terry Ritter is a registered Professional Engineer, a member of IEEE and ACM, with a background in computer architecture, hardware, software, and now, library research. He has spent the last few years being Blue Jean Software and Blue Jean Computer Engineering.

Terry Ritter, his current address, and his top page.

Last updated: 1996-01-04

PUBLISHED: Ritter, T. 1991. The Politics of "Software Patents." *Midnight Engineering*. May-June: 29-35.

The Politics of "Software Patents"

Terry Ritter

What happens when patents apply to software? The somewhat hysterical article "Software Patents" by "The League for Programming Freedom" (in the Nov. 1990 issue of *Dr. Dobb's Journal*) might lead one to believe that we have found the end of software development as we know it. Yet other people, given mostly the same information, see the application of patents to software as a natural and welcome improvement to the current state of software secrecy.

"The League for Programming Freedom" has drawn the "software patents" argument as the freedom of programmers versus oppression by large software corporations. This argument is flawed in many ways, but perhaps the worst part is to assume that, if not for patents, most important software concepts would be free for use. We already know this is false: Important software ideas are now kept *hidden* in trade secret source code. Large software companies *do* have access to trade secret information because they have the resources necessary to disassemble all important systems, but individuals *do* not have similar resources. Thus, trade secret source code is a way for large software companies to share important techniques among themselves, to the exclusion of individuals and small companies. (For example, consider the early years of DOS.) In contrast, a patent is an economic tool which encourages the *disclosure* of new ideas.

I intend to show that "The League for Programming Freedom" has presented a biased argument which supports a particular point of view with selected facts. Observations they did not make include:

1. The proposed alternative to patents is trade secrecy, and this is worse;
2. Protection of software is necessary to protect some hardware mechanisms;
3. Patents encourage research by individuals and small businesses (as opposed to large businesses and bureaucratic educational institutions);
4. Hardware patents are a fact of life for hardware design, and they have not destroyed the hardware business;
5. A patent is virtually the only tool that will allow an individual or small business to compete directly against a large, established software business.

Protecting Software Is Necessary To Protect Some Hardware Designs

For the past couple of years I have supported, at my own expense, my own research and experimentation in the field of cryptography; I have published several papers and applied for patents on the results. On December 25, 1990, U.S. Patent 4,979,832 titled "Dynamic Substitution Combiner and Extractor" issued to me. This invention is a logic mechanism which may be realized in hardware or software, so I made it clear in the specification and claims that the patent does apply to a software implementation. If the patent would not protect against software infringement, few implementors would hesitate to realize the mechanism in software. Without software protection, I have no protection at all.

Of course, my patent may not really be the sort of "software patent" that "The League for Programming Freedom" is protesting. My invention is *not* confined to software, since it would make a fine integrated circuit. My invention is *not* just an improvement in detail, but instead is fundamentally new technology. And a general lack of "prior art" solutions for a well-known problem clearly indicates "unobviousness" in this case. But if "The League for Programming Freedom" is protesting some *other* type of patent, perhaps they will agree that patents like mine, which *must* apply *both* to hardware *and* software to obtain meaningful protection, are necessary and perhaps even desirable. This would mean that someone would have to draw a distinction between one sort of "software patent" and another.

Much confusion seems related to misunderstanding exactly what a patent is. In particular, "The League for Programming Freedom" gives us a number of instances in which "commonly used software techniques" were patented. They say that these patents were upheld *despite* testimony that the techniques had been developed years before. These events are not contradictory.

How Can "Commonly Used Software Techniques" Be Patented?

Most of us probably think that a patent is given to the first person to invent something, but this is not quite correct. A patent may be seen as a *reward* for the *first public disclosure* of an invention. Certainly, any inventor has the right to keep an invention secret, but those who do this also give up even future rights to the patent. By keeping the invention secret, they also risk the possibility that some other inventor may come up with the same invention, patent it, and apply the patent to the original inventor.

The problem with "commonly used software techniques" is that we may see their presence on a display screen, and yet still not know how the trick is done. The true description of an invention is *not* what it does, but *how it does it*, and in software systems, most such descriptions are kept hidden in secret source code. But an invention which is kept secret not only does not qualify for a patent, it also does not constitute a full public disclosure, or "prior art," which would inform the public, and also prevent someone else from getting such a patent. The problem is that many software inventions have been deliberately hidden in tightly-held source code instead of being patented or publicly disclosed.

What Is Wrong With Secret Source Code?

By keeping our source code secret, we distribute the results of our work, but we fail to share *the work itself*, our actual insights into programming. Eventually, every product gets old and is retired, but the source code is still not revealed; the programming insights in it are thus lost forever. It is no wonder that programmers continually "re-invent the wheel," *we have no choice*! Certainly, *a lot* of software is distributed in source code form. But, just as certainly, *very few economically important* systems are distributed in source code, unless written trade secret covenants have been concluded. Obviously these people believe that they have something important to hide. Maybe they are right.

Hidden innovation which is eventually lost to society is *exactly* the situation which patents were intended to correct. In a sense, patents are an indication that society expects us not only to contribute *products*, but also to disclose our *technology*, and not just keep it hidden. Certainly, if a business folds completely, we do not want its accumulated expertise to be lost as well. The application of patents to software may eventually provide enough protection so that we can display source code, even for economically important systems. For the present, the problem is *not* that patents are too *effective*; the problem is that patents do not provide *enough* protection to encourage the public display of source code.

But Isn't Software Invention "Commonplace"?

"The League for Programming Freedom" would have us believe that software invention is common; certainly, *mediocre* software is all too common. However, it seems obvious that an experienced, talented programmer generally produces superior programs. Why should the better individual *not* be able to claim that his or her work is "exceptional" and "superior"? And if we support this, how can we say that such work is not "rare and precious"?

A patent is a non-renewable 17-year legal monopoly on the manufacture, sale or use of an invention. Some think that a patent can be obtained on *any* solution to *any* problem. Such a solution could be *called* an "innovation" or "invention," but few of these would qualify for patent protection. To qualify for patent, an invention must be:

1. In one of five statutory classes (a process, a machine, a manufactured article, a composition of matter, or a new use of one of the other classes),
2. Useful,
3. Novel, and
4. Unobvious.

To some extent each requirement is subject to human legal judgement. But these judgements are not *arbitrary*; for example, there a number of factors which may *indicate* unobviousness, and these can be formally examined. Such factors include "unexpected results," "previously assumed insolubility," "long-felt and unsolved need," "new principle of operation," and so on. By examining the evidence of the "prior art" (previous patents and publications) with respect to the various factors of unobviousness, a reasonable and defendable legal *opinion* can be developed.

When we look at the requirements for obtaining a patent, we see that unobviousness and novelty are unlikely to be satisfied for most software innovations. This, of course, makes patentable software inventions all the more precious.

Yet another source of confusion is "the clarity of hindsight." *After* a problem has been solved, the solution is -- surprise! -- *obvious*. Using exclusive-OR to display a cursor on a computer screen may be obvious *now*; it may not have been nearly so obvious when such screens were rare and new. But the worry over this patent is just one more indication that true innovation is rare and precious: Apparently, "obvious" alternatives to the exclusive-OR cursor are not all that "obvious."

How Can Patents Apply To Fundamentally Different Computer Programming?

The idea that software is "fundamentally different" from hardware is normally a harmless conceit. But when this idea is proposed as a basis for public policy, it deserves considerably more attention.

Software is basically a convenient way to customize a generalized logic machine which was deliberately designed to be so customized. Software is not magic; it can only *select* among the hardware functions which were originally designed into the physical machine. A strictly hardware machine could be built -- without software -- to produce the same effects as any particular program; but **without hardware, software can produce no effect at all**. When it comes to results, software is not *equal* to hardware, but is instead *strictly dependent* on hardware.

So called "low level language" or assembly-language software consists mainly of symbols or specific words, each of which directly selects a particular operation for the physical machine to perform. Such software can scarcely differ in intent from the sequencing performed by a piano roll, a music box cylinder, a clothes washer selector or dishwasher selector, or any other machine sequencing device which is normally considered a part of the machine.

So called "high level language" software consists mainly of *names* for sequences of machine operations; controlling such a sequence is normally the task of a sequencer, which is clearly part of a machine. It does seem clearly unreasonable that simply *naming* a sequence of operations, and then using that name in a lexical work somehow makes the sequencing operation *not* a part of the machine.

Virtually every criterion which has been advanced to establish differences between hardware and software simply establishes a similarity: Software development is *not* cheap (*good* software is almost unbelievably *expensive* to develop); software performance is *not* and (in general) cannot be mathematically guaranteed (it is too complex); software *does* fail (disturbingly often); software does interfere with other software (e.g., MSDOS TSR programs). It is obviously much easier to reproduce software than integrated circuits, but this means that software needs *more* legal protection, not less. Software can *seem* more ideal than hardware because, by itself, software does not -- and *can* not! -- *do* anything; it is the controlled machine which must deal with reality. And real machines are *often* designed by assuming mathematically ideal mechanical or electrical components.

Software is a useful fiction to describe an area of human endeavor and the business of the customization of the generalized logic machines which we call computers. Certainly, written software is a form of human expression like literature, art, or poetry, and should be protected by copyright. But if software has productive worth, this occurs only *by* and *in* a physical machine, and that software must be present in some machine-readable physical storage. The resulting changes in machine state are the "transformation of matter from one physical state to another" as expected under patent law. Since software must be present in a physical form (generally not as text printed on paper), and produces an effect which is only the equivalent of a machine constructed solely of hardware, software deserves patent protection as an expected part of a logic machine.

Can Software Be Exempted From Patent Protection?

It is easy to understand how software can be used to "replace" hardware: Software customizes a logic machine which was designed to be customized; the customized machine can produce different effects at different times; producing those various effects would otherwise require separate hardware.

But if software cannot infringe patents, then there is substantial motive for manufacturers to move infringing hardware logic designs into software. Given the availability of fast microprocessors, much previously protected hardware might be implemented as software (perhaps even as *microcode*). This would obviously weaken, and perhaps even *destroy*, the currently existing protection for some types of hardware. Thus, the simple proposal that software not be considered infringing may, in practice, imply a radical reform of the entire structure of patent protection.

Moreover, when software replaces hardware, *the entire reason for the existence* of the software is to produce *the same effect* as the replaced hardware. In order to perform its intended function, software must be executed, and when it is, it functions as a sequencer, which would otherwise be considered just a part of the larger machine. If software is just a part of the machine, why should it not be protected by patent?

Suppose I Don't Want The Patent . . .

. . . is there a way to prevent *someone else* from getting it? Sure. All you have to do is to make the invention public. A year after you publish a complete description of an invention, it should be firmly in the public domain. (Of course, if someone applied *before* a year after your publication, and could document that they had made the same invention *before* your publication, they might still obtain the patent.) Remember that you must disclose how the thing is done: In software, this generally means *actual working source code*. But even source code, by itself, may not be sufficient for disclosure; documentation and a discussion of the particular techniques probably will be required as well.

Publication does *not* necessarily mean that you need to write an article or paper and get it printed in a periodical. Generally speaking, "publication" means an open and widely available public disclosure of the way the invention works. In the present day and age, disclosure on a computer network or even a bulletin board system may qualify, although this would be a topic for legal council. You should arrange for witnesses who could testify as to the date of public disclosure.

When Writing Software, Do I Run The Risk Of Patent Lawsuit?

No. Most conventional software techniques were publicly disclosed many years ago, and are in the public domain. For example, one should expect that most of the procedures in Knuth's *The Art of Computer Programming* series are freely available for use (with RSA public-key encryption being a notable exception). Various other books and journals disclosed most of the general techniques long ago. Any new patents which would affect software would have to be something rather unusual.

And a patent is an *economic* tool. Although it *could* be applied to individuals, the whole point of a patent is to control *economically significant* infringements. Individuals generally do not make worthwhile economic targets.

Of course, when you *manufacture and sell* software, you do run some risk of lawsuit. But in the day-to-day operation of business, lawsuits are not uncommon; a patent lawsuit is just one risk among many. If you are willing to accept the *possibility* of lawsuit, you probably should not be in business at all. If a patentee believes that you are infringing a patent, you will be informed; in many cases it will be more profitable (for both sides) to negotiate some sort of license agreement than to proceed to court. In other cases, you may have to re-design your product. Very, very few patents are so powerful as to prohibit any other way of producing the same result.

But if you basically have the idea that you can do anything you want, that nobody can tell you what to do, and that you will not pay anybody anything, a patent lawsuit may be the least of your problems.

Can I Eliminate The Risk Of Infringement By Performing Patent Searches?

Alas, no. Patent searches are both difficult and unreliable, although this situation is hardly unique. Real estate, for example, is often encumbered with various sales of rights and liens. Real estate title searches are complex and unreliable, making title insurance necessary.

Most of us are located within a reasonable distance of a patent deposit library. It is certainly *possible* for an individual to perform an in-depth patent search in such a library, although this is an arduous task. The first time you search a particular topic, expect to spend at least *several full weeks* at it. The main difficulty is the process of physically finding and retrieving the patent abstract (among huge shelves of such abstracts), or the patent itself (one among hundreds in a film taken from a wall of microfilms), along with the large number of patents which generally must be inspected. Because actual paper patents are physically collected by class at the Patent and Trademark Office in Washington, D.C., an equally effective search might be made there in only a day or two.

Although technology may eventually reduce the difficulty of a patent search, such a search is always going to be imperfect. A searcher must judge essential similarity based on only few minutes of reading a complex patent; some of those judgments will be wrong. And, in any case, one can only research *granted* patents; pending applications are secret until they issue.

Software Is Essentially *Information;* Isn't Information Free?

When we are young and in school, information is provided for us, and the vehicles of such information are freely available. But as mature individuals we must understand that we are part of a *capitalist* society, and *all* of the information we have was found, accumulated, *and paid for*, through the direct effort and expense of previous generations. It seems a bit too clever to assert that "information is free" when it was paid for by someone else. Even libraries must purchase books (which were published to make a profit), then store them at huge expense; the fact that such expenses are hidden from individual library users does not mean that the expenses do not exist. Libraries are not free.

Whenever information is free for use, it is because someone who found or created it (and who thus "owned" it) decided to make it available to society at large. Probably the owner released the information to in some way make a profit (perhaps to publish a book or enhance a reputation), or perhaps this was just the expected return for an academic salary (which is certainly an individual profit). Encouraging inventors to *reveal* their work (in return for a limited term monopoly) is the whole point of the patent system.

Many software ideas are available to users. But other software ideas have not been made public, and are instead kept secret. By applying the patent system directly to software, we may be able to make some of these ideas available for a negotiated fee in the near term, and for free use by the next generation. Maybe that generation will not have to continually "re- invent the wheel," but can instead get on with extending the limits of software design.

Why Encourage Software Invention?

One aspect of patents is that they can provide an economic basis to support research and development. Patents generally provide rewards only when they are *actually applied* in a product, when that product is *actually produced*, and when it *competes successfully* in the marketplace. Patents return royalties only when research is actually translated into results; patents thus cost society virtually nothing, unless and until buyers freely pay for an improved product. Compare this with the usual bureaucratic approach of government sponsored research.

Another important aspect of patents is that they provide a way to capitalize the invention. New businesses are extremely risky. A new business with a strong patent position is in a better position to acquire capital and build the strong organization needed to become an effective competitor. The results are that the consumer gets innovative products, from a stronger company, which may provide more competition in the marketplace. This is what we want.

In some sense, ideas *are* free. But the time and effort involved in verifying and improving the idea into an invention, reducing it to practice and testing it, then formally reporting on it; these things are certainly not free. Recognition of the importance of a new invention, comparing it to previous inventions, and extending old theories to fit the new situation are also not free. Designing a product using a new invention, building the product, finding and informing buyers, and selling the product -- all this is not free! And a successful product may have to fund many other less-than-successful developments. If some other company can simply *copy* your original "free" idea *after you* have invested in bringing it to the market, you are not going to do this very often. Such a situation is not what we want.

Don't Patents Prevent Small Businesses From Competing?

Patents are a limited-term monopoly, and they can be used to prevent *any* size of business from producing a competing product. But there is not much point in applying a patent against a tiny business which is neither an economic threat nor a significant source of royalties. Thus, patents are most *effective* against large businesses.

In the present software business environment, small companies *already* cannot compete directly against a large company. The advertising, production and marketing resources of a large software firm can make even dull products dominant in the marketplace. And because a large software company generally has more programming resources, their product may even be better designed, with more features than one produced by a small company. One of the few things which might allow a small firm to compete directly against a large one is a strong patent position.

Don't Patents Hinder Development Of "Public Service" Software?

"The League for Programming Freedom" cites the "X Window" project as an example of "public service" software which may infringe several patents. They say "dozens of companies and hundreds of thousands of users who accepted software from MIT with the understanding that it was free are now faced with possible lawsuit."

Note that this is the very *same* MIT which in 1983 received U.S. Patent 4,405,829 for the RSA Public Key Cryptosystem, a patent which is generally considered to cover software implementations. That a software project connected with MIT is itself inconvenienced by patent protection would seem to be simply a matter of everyone having to play by the same rules. Should we now be able to freely distribute RSA programs, as long as they may be a "public service"?

The real question here is why the X Window software was not quickly modified so that it did not infringe. If "public service" software projects do not have sufficient resources to work around issued patents, perhaps they are not as much of a public service as they claim to be. The horrible truth about "free" software is that, without continuing support, the software is often more trouble than it is worth. But only the continuing *business* of software is financially able to provide such support.

Why Should The Software Business Get Special Treatment?

Although patents may be new to the software industry, the need to respect intellectual property is old indeed in most forms of manufacturing. Modern computer manufacturers must contend with patents as a matter of course, yet they still manage to produce many new designs. In most cases, competition and general economic conditions are *far* more of a threat to the survival of a manufacturing business than petty patent disputes. Now software patents make a few waves, and the call goes out to exempt the software business entirely. It does not make sense.

Conclusions

First, the main reason that "commonly used software techniques" can be patented is that these techniques have been *selfishly retained* by their inventors as trade secrets. Not only were the techniques not patented, they also were not made available for public inspection and use. It does seem a bit hypocritical to argue that society should not enforce patent protection for those who would reveal their techniques, when society has already supported trade-secret protection for those who would hide them.

Second, the range of patents which must necessarily cover software implementations may be *far larger* than "The League of Programming Freedom" expects. And it may be difficult, both legally and economically, to differentiate such software from the pure hardware logic machines (e.g., computer hardware itself) which we do expect to enjoy patent protection. A decision that software could not infringe patents might effectively eliminate protection for a large class of logic designs, including computer hardware.

Third, if good programming is actually "rare and precious," then (providing it fulfills the various requirements for a patent) it also *deserves* the historic reward for the disclosure of new techniques. An invention is *not* just an idea, it also includes reduction to practice, testing, and formal disclosure, and may well include extensive research and development as well. Even *recognizing* a patentable technique can be an important service! Unless we can argue that all software concepts are necessarily trivial (thus requiring no research, experimentation or testing), society *should* support the marketplace recovery of software development expenses and profit.

Fourth, although the process of recognizing that software can infringe patents may be difficult, the difficulty should not be overly exaggerated. Hardware businesses have always had to recognize patents, yet they manage to survive and prosper. Most patented techniques have alternatives, and sometimes the alternatives are actually superior. It is a sign of the importance of the software industry to society that we must now begin to recognize, record, and reward true advances in our technology.

And, finally, a patent provides one of the few tools which might allow a new business to compete in a mature industry. The natural application of patent concepts to software has the potential for improving the situation for individuals and small businesses, by allowing them to restrain the giants. Of course, large companies may see this as a threat. Perhaps they will even support "The League for Programming Freedom" to try to keep this threat at bay.

References

1. Pressman, D. 1988. *Patent It Yourself!* Nolo Press, 950 Parker St., Berkeley, CA 94710, (415) 549-1976.
2. Samuelson, P. 1990. "Should Program Algorithms be Patented?" *Communications of the ACM*. August. 33(8): 23-27.
3. Schoppers, M. and R. Schmidt. 1991. "Patently Obvious" (ACM Forum). *Communications of the ACM*. August. 34(2): 81-82.
4. "Software Patents." 1990. *Dr. Dobb's Journal*. November. 56-73.

About The Author

Mr. Ritter is a registered Professional Engineer, who is also known as Blue Jean Computer Engineering and Blue Jean Software. He built a working "Mark 8" 8008-based computer in October, 1974. He has experience in both hardware and software, having done microprocessor design for Motorola, and written the software for a dedicated multi-processor local area network for industrial control applications. He has recently concentrated on the design of cryptographic systems.

Terry Ritter, his current address, and his top page.

Last updated: 1996-05-28

PUBLISHED: Ritter, T. 1991. Transposition Cipher with Pseudo-Random Shuffling: The Dynamic Transposition Combiner. *Cryptologia*. 15(1):1-17.

Transposition Cipher with Pseudo-Random Shuffling: The Dynamic Transposition Combiner

Terry Ritter

ADDRESS: Blue Jean Software, 2609 Choctaw Trail, Austin, Texas 78745.

ABSTRACT: Extensions are made to a class of transposition cipher based on continued shuffling. These ciphers permute plaintext into ciphertext by swapping every message element with some message element selected at pseudo-random; elements can be characters (e.g., bytes) or bits.

Extensions include operation on very large data blocks, cryptographic shuffling variations, and the efficient extraction of plaintext from ciphertext. In addition, selected extra data can be adjoined to the plaintext to eliminate the data-dependent weak encipherings otherwise inherent in transposition. This bit- balancing data is supposed to completely eliminate all normal usage-frequency statistics from bit-transposition ciphertext.

The same mechanism can also be viewed as a cryptographic combiner, and, with sequence-to-block and block-to-sequence conversions, can generally replace the exclusive-OR combining function used in Vernam stream ciphers.

KEYWORD LIST: cryptography, computer cryptography, cipher, block cipher, permutation, transposition, dynamic transposition, combiner, cryptographic combiner, mixer, shuffle, data balancing, bit-balancing

---

INTRODUCTION

This paper extends an existing cryptographic mechanism which can be described as dynamic transposition. This mechanism *combines* two data sources into a complex result; one data source is accumulated into a block, and the other is used to re-arrange that block. A related inverse mechanism can *extract* the accumulated data from that result.

Any form of transposition would seem to require some accumulation of data. Since data accumulation and serialization are easy with modern technology, dynamic transposition can be used to replace the Vernam exclusive-OR combiner in stream ciphers. The various techniques used in Vernam ciphers can also be applied to dynamic transposition; any cryptographic advantage of the resulting cipher is thus due to the additional strength of the new combiner.

This paper develops a particular form of dynamic *transposition*; a related paper develops a form of dynamic *substitution*. Advantages of the transposition form include an interesting level of secrecy in the resulting ciphertext and the virtual elimination of meaningful statistical analysis measurements. These advantages are purchased at the cost of some inherent increase in processing effort, and the need to encipher data in blocks instead of individual characters.

BACKGROUND

For a general background in cryptology see Kahn [14], and for details on the classical systems and their analysis see Gaines [12]. More modern statistical approaches are given by Sinkov [26] and Deavours [7]. A good partly-technical anthology is Deavours *et. al.* [6]. There is a nice but older survey by Mellen [19], a major effort by Diffie and Hellman [9], and a newer one by Massey [18] (also see the other papers in that issue). A rigorous but not always applicable theoretical base starts with Shannon [25] and is extended by Hellman [13]. A relatively modern technical reference is Beker and Piper 1982 [1], although the generally more introductory Beker and Piper 1985 [2] is a major reference for this paper, and will henceforth be referred to as B&P 85. Denning [8] and Pfleeger [23] present cryptography in the broader context of computer security issues.

TRANSPOSITION

A *transposition* cipher re-orders or *permutes* plaintext elements into ciphertext [12, 26]. If a single transposition can be thought of as a simple exchange in the positions of two elements, it is the simplest form of permutation; moreover, any possible permutation can be constructed (in many different ways) with sequences of transpositions. Permutation has been used for entire ciphers (mainly in an era of pencil-and-paper operations), and, in a limited form, is still in common use inside substitution-permutation networks [11] of the sort from which the U.S. Data Encryption Standard [e.g., 21] is built.

Most previous descriptions of transposition or permutation ciphers have generally concerned fixed or *static* permutations. However, B&P 85 [2] does give the basis for *dynamic* permutations, in the sense that each overall permutation is newly created (one transposition at a time) by a continuing pseudo-random sequence. (To some degree, the paper by Costas [3], and comments by Klingler [15] and Costas [4] anticipate some of this work.) Although not stated in B&P 85, this means that *every* block is likely to be permuted differently, no matter how many blocks there are (within reason). Moreover, dynamic transposition mechanisms can be made to handle very large blocks, as well as dynamic changes in block size.

SIMPLE DYNAMIC TRANSPOSITION

B&P 85 [2, p. 97] and Klingler [15] describe enciphering blocks using the well-known data *shuffle* algorithm [10, 16, p. 139]. The shuffle process steps through a block of data, element-by-element, and exchanges each element with some element selected at random. In this way, any original element can wind up anywhere in the block. In one pass, any particular element is guaranteed to be exchanged at least once, is probably exchanged twice, but may actually be exchanged more often. Some sort of pseudo-random *confusion* sequence is, of course, needed for each exchange operation.

The execution time of the shuffle process (in a software implementation) is generally proportional to the number of elements being shuffled. Consequently, the shuffle algorithm gives us no particular reason to encipher plaintext in units of small blocks. And, since the complexity of the result would seem to increase with the number of elements in any particular permutation, there is a strong motivation to use large blocks. Indeed, in most cases, each message could probably be enciphered as a single large block, or even a sequence of variable-size, yet sizable, blocks; the shuffle process takes about the same amount of work however organized.

TRANSPOSITION PROBABILITIES

Mathematically, a cryptographic transposition process generates a *permutation* of the input data; that is, the data are simply re-arranged. The shuffle algorithm is a convenient way to construct one of the many possible re-arrangements at random. How many possible arrangements are there? Suppose the block has n different elements; the first element can be positioned in n possible places, the second in $(n - 1)$, the third in $(n - 2)$ and so on, for a total of $(n)(n - 1)(n - 2)...(1)$, or $n!$ (n factorial) possibilities. So the probability of getting the correct deciphering at random would seem to be 1 out of $n!$. This is very encouraging, since factorials can yield some really remarkable values. For example, a 64-element block would be considered rather small, yet the probability of correctly deciphering such a block at random would seem to be 1 out of 64!, or about 1 in 10^89.

Unfortunately, the usual situation is somewhat more complex, since a data block is not constrained to have just *one* occurrence of any particular data value. But when there are multiple occurrences of the same value, it surely cannot matter which of those goes in which position when deciphering. So multiple reverse permutations will each generate a correct deciphering (though most of these will yield no information about how the block was permuted). There are $k!$ different permutations which produce the same deciphering for k occurrences of any particular value. Consequently, there are $(k1!)(k2!)...(ki!)$ equivalent decipherings, for ki occurrences of each value. So the probability of getting one of the correct decipherings is the product $(k1!)(k2!)...(ki!)$ out of a total of $n!$ possible decipherings (for block size n). Note that all but one of the correct decipherings represent an incorrect permutation, so even if the correct deciphering is known, finding the particular permutation involved should be exceedingly difficult.

Suppose there are 26 different characters in a 26-element block; there are 26! (about 4 x 10^26) different ways to permute that block. Since each element is different there can be only one correct deciphering, so there is only one chance in 10^26 of finding this permutation at random. But if the 26 characters in the block are *all the same value*, no permutation of any kind will cause any apparent change. Accordingly, there is no way to hide a block of similar data values with a pure transposition cipher.

The realization that *the cryptographic strength of transposition depends upon the data to be enciphered* is both an unexpected and serious complication. It seems only reasonable that a cipher should be able to protect any possible sequence of plaintext data. For example, one user may wish to encipher computer machine code, and another may wish to encipher graphics images. Such computer-oriented data may be very complex, yet still contain long sub-sequences of identical values. It is up to the cipher system to handle these sequences in a strong manner. Classical transposition cannot do this.

SHUFFLING PROBABILITIES

From one point of view, the shuffling process converts a confusion sequence into an enciphering permutation. We know that there are $n!$ such permutations in an n-bit block, but how many confusion sequences are there?

The confusion sequence must select one of the n block elements as an "exchange partner" for each element in the block. Accordingly, there are n possibilities for the first partner, n again for the second, and so on, for a total of $(n)(n)...(n)$ or n^n possible different selection-sequences. But there are only $n!$ possible permutations, so each enciphering permutation is created by $(n^n / n!)$ different sequences, on average.

Suppose we are shuffling that same 26-element block. We need 26 pseudo-random values, each of which selects one of the 26 possible block elements; there are 26^26 such sequences, about 6 x 10^36 of them. But there are only 26! enciphering permutations (about 4 x 10^26), so about 1.5 x 10^10 (15 billion) different sequences will create each permutation. So, even if the correct permutation is somehow known, finding the particular pseudo-random sequence which created it should be exceedingly difficult.

A DYNAMIC TRANSPOSITION CIPHER

Consider a byte-shuffling block cipher: The plaintext will be collected into a block, then a controller will walk through the block, byte-by-byte, exchanging each byte with "partner" bytes selected by a random number generator. For cryptographic purposes it may be reasonable to generalize the shuffle process: For example, it is unnecessary to visit bytes in sequential order, nor need the shuffling stop after exactly one pass [15], as long as the deciphering system follows similar rules.

Letter (byte) frequency statistics are obviously unchanged by the shuffling process. But the frequency distribution statistics do not seem to aid deciphering nearly as much as they would on a simple substitution cipher. Block transposition ciphertext might be compared to a set of alphabet tiles: A particular message can be built from those characters, but once they go back into the storage box, how can anyone decide what particular message they once represented? Indeed, unordered letters on their own *cannot* represent *any one particular* message; instead, they stand for *all the possible* messages which can be made from them.

Actually, since the message is expected to be a correct grammatical example, with correctly-spelled words, on an expected subject, which exactly covers the ciphertext letters, cryptanalysis may not actually be impossible. The normal cryptanalytic technique for transposition ciphers consists of obtaining two messages of the same length, both of which are presumably permuted in the same way. By anagramming both messages in the same way, the correct permutation is found when both messages read clearly [14, pp. 225-226]. Some assistance is available in the form of letter digram and trigram counts, which can support a particular inverse transposition by indicating the presence of statistical plaintext [13, p. 402]. But dynamic transposition need not generate any similar permutations, even for consecutive blocks of similar size.

Because a character or byte transposition combiner can provide good performance only when given a block containing different values, it could be useful to place a Vernam system (an exclusive-OR operation with a pseudo-random stream) before the transposition data input. In this way the plaintext data could be "randomized," and thus need "never" produce a stream of identical values which would compromise the strength of the transposition encipherment.

DECIPHERING DYNAMIC TRANSPOSITION

B&P 85 [2, pp. 93-96] describes a multi-step process of explicitly defining the permutation, finding the inverse, and then un-permuting the block according to the inverse permutation. It may be surprising that there is an easier way.

The shuffling process destroys no data; elements are just repositioned. The values involved in any particular exchange could easily be replaced, simply by exchanging them again, if those values had not been moved by later processing. So the last pair exchanged can be exchanged back into their previous positions. Once that pair has been replaced, the next previous pair can be undone, and so on. Thus, to decipher the shuffled data, the exact same element pairs need only be exchanged in reverse order.

During enciphering, the exchange pair is generally selected by a counter or other process, and a pseudo-random value. It is easy enough to run a counter in reverse, or the desired number of values could be collected in a buffer, and then simply accessed in reverse sequence; the pseudo-random sequence can be "reversed" in the same way. This provides all the information needed for deciphering. (In practice, very long sequences can be accommodated by writing filled buffers to a disk file; the file- stored buffers can easily be recovered for use in reverse order.)

BIT-LEVEL DYNAMIC TRANSPOSITION

In the same way that letters or bytes can be shuffled, individual bits [2, p. 93] can also be shuffled. In this way the elements of any particular character or byte might be spread throughout the ciphertext. Since any particular bit looks remarkably like any other, how is a cryptanalyst to select those bits which belong to any particular plaintext byte? Of course, the cryptanalyst does not have to find the particular bits corresponding to a *particular* byte, since any bit of a given value is exactly the same as any other. But this also means that there can be no way to tell which bits belong together.

Byte-level frequency statistics are destroyed by bit-level permutation; only bit-level statistics are left. The cryptanalyst can know how many ones and how many zeros there are in the block (these are the same as in the original plaintext), but this does not seem to help much. Since virtually any message can be constructed from those bits, how is the cryptanalyst to select the correct one?

One interesting implication of bit-level exchange operations is that they are often ineffective. When byte values are being exchanged, the exact same value is "exchanged" (for zero net effect) about one time in 256 (assuming an even distribution). But, when working on bits, the other bit is exactly the same about half the time, for no net effect, and when the bits are different, exchanging them simply changes both bits. Even though half of the exchange operations will have no effect, the number of effective bit-changes is still on the same order as the number of bits (two bits change on every effective exchange). And if this turns out not to be enough, each block could be additionally shuffled, perhaps twice or more. In the end, some bits may always remain unchanged, others will be changed, while still others will be changed and changed back again.

SMALL-BLOCK PROBABILITIES

Suppose we continue working with that same small block of 26 character-elements, each of which we assume to have 5 bits (perhaps a Baudot coding); thus the block contains 130 bit-elements. There may be multiple occurrences of some characters in the block, but for a bit-level analysis this is irrelevant. Suppose we have an even number of ones and zeros (the best possible case), 65 of each: Since any one-bit could substitute for any other one-bit, and any zero-bit substitutes for any other zero-bit, there would then be (65!)(65!) deciphering permutations, out of the 130! possible.

The direct evaluation of expressions like these is far beyond the capabilities of a scientific calculator or most computer languages. But it is possible to build such numbers in logarithmic form, and once into logs we use addition and subtraction instead of multiplication and division. For the factorials 65! and 130!, we want the sum of the logs of the integers 2 through 65, and 2 through 130, respectively. There are approximations for the log factorial, but with a computer (and for the values here), it is probably about as easy to sum the logs explicitly.

By actually doing each log and the sum we get ln(65!) = 209.34, and ln(130!) = 506.13, approximately. These values are exponents or powers of *e* (the base of natural logarithms), and would lead to results too large (or small) to evaluate. It seems reasonable to change to the more-familiar base 2, so that we can think of these huge values as the number of bits it takes to represent them. Dividing by ln(2) = 0.6931, we get log2(65!) = 302.01 and log2(130!) = 730.19; these exponents thus represent some binary values which are 303 and 731 bits long, respectively.

For the 130-bit block, 130^130 or about 2^913 possible confusion sequences will generate 130! or 2^730 possible enciphering permutations: The number of possible permutations is huge, and this hides the plaintext. About (65!)(65!) or 2^604 different permutations will encipher a plaintext block into a particular ciphertext block: The number of deciphering permutations is huge, and this hides the correct permutation (even from known plaintext). An average of 130^130 / 130! or about 2^183 different sequences will create any particular permutation: The number of possible sequences is huge, and this hides any particular sequence (even from a known permutation). Thus, the classical attacks of brute force and known plaintext would seem to be wrong ways to penetrate dynamic transposition.

A seemingly different approach would be a bit-by-bit defined-plaintext attack, since this might (if the rest of the system does not prevent it) succeed in building up a complete description of a particular enciphering permutation. Of course, this would mean that the cryptanalyst had that plaintext already (indeed, was *generating* the plaintext), so the attack would be on the pseudo-random sequence. But 2^183 different sequences could have created that permutation (and those sequences are distributed among 2^913 possible sequences), so there would seem to be no way for a cryptanalyst to select the correct one.

THE EFFECTS OF BLOCKING

If all data to be enciphered and deciphered are already in the form of blocks, then each block is (obviously) already full and can simply be handled as a unit. But whenever variable amounts of data are to be enciphered as blocks, the last block is unlikely to be completely filled, and the unused area must then be filled or *padded* [21] with extra data. On average, half a block of padding is required for each message, thus expanding the ciphertext; this is a motivation for limiting the block size. This may not be a particularly significant motivation, however, considering the amount of data which may be easily stored and quickly communicated by computer, and padding is unnecessary when variable-sized "blocks" are available.

A more significant complication is that any padding must be in some way distinguished from the plaintext data, so that it may be removed in deciphering. In general, a particular data value cannot be used as a separator, because "binary" data (such as computer object code) may be enciphered, and such data may contain every possible byte value. The conventional solution is to include some sort of length value along with the padding, which is then removed with the padding when deciphering.

Another complication of data blocking, at least for dynamic transposition, is that the block size defines the value range needed on the "random number" input (as well as the number of values required). Thus, if dynamic transposition is to accept variable block sizes, the random number range must be able to cover an arbitrary block size. And even fixed size blocks, if they are large, can demand a fairly large random number range. For example, a 256 kilobyte block contains 2,097,152 bits, which implies a 21-bit random value to select between them. Shuffling that block requires 2,097,152 of those 21-bit values (and about 6 megabytes of temporary disk storage to reverse that sequence when deciphering).

At first glance, it seems reasonable to pad with random data, since this should help to obscure the data in the block. This idea can be extended: Instead of completely filling each block (except the last) with message data, each block can instead be partially filled with plaintext data and then padded with random data [22]. Naturally, this causes some ciphertext expansion, but the random data should help to dilute the remaining bit statistics, and bit statistics seem to be the only statistics left.

STATISTICALLY-FLAT CIPHERING

But instead of just hoping that the random data will smooth out the bit statistics, steps can be taken to guarantee this result. In particular, the number of one-bits and zero-bits in the plaintext data can actually be counted. Then the message can be extended (or a block filled out) with non-random data so as to balance the bit distribution *exactly*. (Of course we might deliberately vary the bit-balance somewhat from block to block.) After bit-shuffling the extended message, there seems to be very little statistical frequency information left: no word statistics, no letter statistics, and no bit statistics. If some statistical relationship remains which might assist in entry, it is certainly not clear what that might be.

With data balancing, the best possible situation for a transposition cipher can be *guaranteed*. Blocks which might be all ones or all zeros can be balanced and enciphered in a strong way; without balancing, it would be impossible for transposition to provide any effective enciphering of such blocks. And, while the normal block (not heavily weighted one way or the other) may not seem to need additional strength, such blocks also require only a minimum amount of balancing.

Bit-balancing does cause some ciphertext expansion (perhaps 25% to 33% on text files). Naturally, this expansion could be mostly eliminated if the input data had an even bit distribution, and a good distribution might be enforced by passing the data through a Vernam system before transposition. Alternately, modern data compression processing can reduce the size of typical text files by an amazing 60% while simultaneously improving their value distribution. Subsequent expansion due to final bit-balancing should be less than 10% of the resulting smaller file. Thus, if data expansion is a problem, that problem can be managed (at some expense); in many cases, a modest amount of data expansion is not a problem.

The fact that the strength of the transposition can now be guaranteed, independent of the input data, is very significant. Without such a guarantee, it might be necessary to monitor the input to a transposition module and make special provisions for alternate ciphering when strongly-unbalanced data are encountered. With a guarantee of strength, the transposition module can stand alone and handle any arbitrary data sequence before passing it along to another module.

Bit-balancing also provides the basis for an analysis of the strength of the resulting cipher. Since every block is to be balanced, each block should have the same permutation possibilities: one permutation is correct, others are incorrect but still decipher the block, and others are simply incorrect.

STATISTICALLY-FLAT BLOCKS

When working with blocks, there is some difficulty deciding how much space to leave for bit-balancing. A good distribution might need only a little extra data to achieve an exact balance, but some plaintext sequences might be all ones or all zeros, and those would require as much balancing data as plaintext data.

By counting data bits while filling the block, we need only leave space for exactly the number of bytes needed for bit- compensation. But there must be some way to isolate and remove the bit-compensation when deciphering. One way might be to enter bit-compensation data, of the least-used bit type (all-zeros or all-ones bytes), backwards from the end of the block. This continues until the bit counts are within a byte of balance. Then exact balance can be achieved with a single byte containing at least one bit of the most-used bit type. Because the previous balancing bytes have contained only the least-used bit type, the last balancing byte is a *contrasting* byte.

This means that the first byte (from the far end) that is a contrasting value is also the identifiable last byte (from the far end) of the bit-compensation data. Thus, the bit-compensation area can be as large or small as needed, and there need be no special code or count to delimit it. Moreover, all but one of the bits of the minimum two balancing bytes can participate in balancing. Since most data distributions will need at least two balancing bytes anyway, the average overhead for defining the balancing data area (beyond that required for simple balancing) would seem to be *less than one byte*. The same technique can be applied to huge or dynamically-variable blocks, and some added computation can produce similar results for complete message permutations.

All fixed-size-block ciphers which support variable-length data need a mechanism for padding the last block. But if bit- compensation is already supported, it is possible to bit-balance the filled portion of the last block, and then complete the block with particular bit-balanced byte values. After deciphering, proceeding from the end of the block, all the particular bit-balanced byte values can be skipped. Then, if there are all-ones or all-zeros bytes they can also be skipped, along with the next byte (which is the contrasting byte). In this way, the same mechanism which is used to delimit the bit-compensation can also remove the padding at little or no extra cost.

NUMERICAL SECRECY

For a modest block of 512 bytes by 8 bits (the size of a single MSDOS logical disk sector) the block size is 4096 bits. (In actuality there will be a minimum of two balance bytes, so there will be at most 4080 data bits, and may actually be many less.) Assuming an even bit distribution (which we can now enforce), there are (2048!)(2048!) decipherings out of 4096! possible permutations. In logs, ln(2048!) = 13571.95, and ln(4096!) = 29978.65; so there are about e^29979 or 2^43250 possible permutations, a 43251-bit binary value, and only one of these permutations is "correct." (In ordinary terms, "many" other permutations would be "readably close," but in comparison to numbers like these, these possibilities pale to insignificance.)

The total number of deciphering permutations is e^27143 or 2^39160, a 39161-bit binary value; so finding the one correct permutation would seem to be a difficult task. And the average number of sequences which create any particular permutation is e^4091 or 2^5902, a 5903-bit binary value. Of course, instead of 1/2 K (byte) blocks, we might well do entire files of 10K, 100K, or perhaps even 350K in length.

These probability calculations have a strange other-world character to them. While the results do imply a sort of fundamental secrecy for the dynamic transposition mechanism itself, they do not imply that a cipher using this mechanism is necessarily secure; any cipher is only as secure as its weakest link. Basically, these results are useful only to say that an exhaustive search of permutations and sequences, for even a modest (correctly constructed) block, is completely out of the question. Then, if no other part of the cipher has an easy "shortcut" attack [21, p. 137], the cipher may be secure in practice.

A simple cipher module like this actually may be much more valuable than a complex one, for it may eventually be possible to understand its exact limitations, and then answer those limitations completely in other modules. Although it is elegant to have a single complex framework handle all aspects of secrecy, such systems usually cannot be completely understood in a deep way. For example, there has been suspicion of a DES "backdoor" for over a decade, and great strides have been made in factoring large numbers like those used in RSA. A reasonable alternative is the selection of simple mechanisms which can be deeply understood.

Note that a shuffle permutation of a 512-byte block requires 512 12-bit pseudo-random values. Thus, to encipher 4096 bits we need 49,152 pseudo-random bits, for a "key stream" expansion of 12:1. Since a Vernam cipher needs only a single random bit to encipher each data bit, shuffling dynamic transposition is seen to be relatively demanding of the random- number resource. But the expansion of this resource may be only a small part of the cost of a complete cryptosystem, and what it buys, hopefully, is cryptographic strength.

When transposing bit-balanced fixed-size blocks--each of exactly the same length and with exactly the same number of one-bits and zero-bits--in some sense *there is only one block*, and all of our different ciphertext blocks are only permutations of that same aboriginal balanced block. Moreover, all of the bit-compensated plaintext blocks and *all possible decipherings* are just other permutations of that same primitive block. Various decipherings include all the possible bit-balanced messages which will fit in the block, including a huge number of cases in which the messages differ only in their crucial words. There would seem to be no way for a cryptanalyst to distinguish the correct message from all possible decipherings. So brute-force methods would seem to be useless, as well as impractical.

TESTS

The dynamic transposition combiner may be considered to be a *black box*, with two input data ports ("Data In" and "Random In"), and one output port ("Combiner Out"). Because the block size can vary, the "Random In" range must also vary. Evidently the mechanism inside the box in some way combines the two input streams to produce the output stream. It seems reasonable to analyze the output statistically, for various input streams.

For these tests, the "Data In" stream was a sizable text file (a book chapter) with all spaces and punctuation deleted, and lower case converted to upper, leaving a 26-element alphabet of 18,135 capital letters.

MEASURES OF RANDOMNESS

The black box test results can be summarized in the form of standard "delta IC" [20], and "Z-coefficient" [7, 17] computations. In both cases, we count the number of occurrences of each element value in the stream being analyzed.

The index of coincidence (IC) is conceptually "the sum of the squares" (of the element counts) "over the square of the sum" (or total count); the IC is normalized to a delta IC by multiplying by the size of the alphabet. A delta IC value of 1.0 indicates a random distribution.

A Phi value is conceptually "the sum of the squares of the element counts," and an "expected" value of Phi and a statistical variance value can be derived for a random data stream. The Z-coefficient is just the difference between the actual and expected Phi values, normalized by dividing by the variance. A value of 0 would be expected, and a value between -2 and 2 would be most probable for a random sequence.

The results are summarized in Table 1.

```
Table 1.

TRANSPOSITION DISTRIBUTION STATISTICS  (delta IC / Z-coefficient)

                             Data In      Random In      Combiner Out
The Data are 26-Letter Text
   Byte Transposition     1.66 / 1684    1.00 / -0.9    1.61 / 1593
   Bit Transposition      1.66 / 1684    1.00 / 1.8     1.32 / 257.5
   Bit Balanced Trans.    1.66 / 1684    1.00 / 0.8     1.00 / -0.2
The Data are One Value Repeated
   Bit Balanced Trans.    26.0 / 36199   1.00 / 1.0     1.00 / 0.8
```

Apparently the bit-balanced bit transposition creates output with good random characteristics, even when the data input is just a repeated constant value. (If the data input is random, then clearly the resulting block must also be random, even if the random input is a constant value.)

THEORETICAL SECRECY

In general, a cryptographic combiner can be expected only to increase (albeit greatly) the complexity of a cryptanalysis. Nevertheless, bit-balanced dynamic bit-transposition seems to have some interesting characteristics.

If a bit-balanced ciphertext block carries information, it does so only in its bit arrangement, and any bit-balanced block can obviously be re-arranged into any other. Since any message we can possibly encipher must produce one or more bit-balanced ciphertext blocks, any ciphertext block can obviously be re-arranged into any part of any possible message; all except one of these is a meaningful false solution, or "spurious message decipherment" [13, p. 290]. Hellman defines the number of spurious message decipherments as *nm*, and writes: "If *nm* takes on large values with probability close to one, then the system will be secure even if the cryptanalyst is allowed unlimited computation." A cryptanalyst would seem to be unable to tell which of all possible message blocks was sent.

Enciphering a block with bit-shuffling implies the existence of some sort of confusion sequence which may itself be penetrated; if the confusion sequence could be analyzed and replicated, the cipher would be broken. In mounting such an attack, the cryptanalyst's first problem would be to determine the correct deciphering permutation. Even an exact copy of the original plaintext block would seem to be of little help: There are a multitude of deciphering-but-incorrect permutations (too many to try them all), with apparently no way to identify the correct one. (Hellman calls this "spurious key decipherment.") The cryptanalyst's next problem would be to identify the particular confusion sequence which produced the known permutation. But since the shuffle process could produce any particular permutation from a host of different confusion sequences, there would seem to be no way to identify the one original confusion sequence so that it might be analyzed. (This would seem to be another level of "spurious key.")

Shannon [25, p. 680], defines *PM(E)* as the conditional probability of ciphertext (block) *E* if message block *M* is chosen, and *P(E)* as the probability of obtaining ciphertext *E* from any cause. If the selected permutation process does indeed map an arbitrary (balanced) block to every possible (balanced) block, it certainly seems plausible that *PM(E) = P(E)*, which is a necessary and sufficient condition for perfect secrecy. That is, if any message block could generate any ciphertext block with about equal probability, then the probability of obtaining any particular ciphertext block cannot depend on the message; Shannon writes, "*PM(E)* must be independent of *M*."

An implication of this is that "the number of different keys is at least as great as the number of *M*'s." In this analysis, the number of "keys" is the number of possible permutations, or $n!$ (for an n-bit block), and the number of possible messages (blocks) is under 2^n, which is far less. It appears that this does not necessarily imply that the number of user-keys must be $n!$, or even 2^n, because the confusion sequence is isolated by the strength of the dynamic transposition mechanism. But, as always, the number of user-keys must be sufficient to prevent a key-search attack.

Of course, a fully-detailed strength analysis probably depends upon a deeper understanding of the shuffle process. Of particular interest is the effect of the confusion sequence on permutation distribution. But the simplicity and intuitive generality of shuffling would seem to bode well for such an analysis, and shuffling is just the basis for this particular form of dynamic transposition.

APPLICATIONS

One use for dynamic transposition would be as a *combining* or *mixing* function for data blocks. With some stream-to-block conversion, and vice versa, such a combiner could be used to replace the exclusive-OR logic function in a Vernam stream cipher. Alternately, it could be used to combine two pseudo-random sequences, to produce an even more complex sequence. Or it could be applied in a product cipher [25] as one part of a chain or network of cipher operations.

Dynamic transposition may be slower, but perhaps also more secure than some other alternatives. Consequently, it might well be used for key delivery as opposed to general data encipherment.

CONCLUSION

Transposition--normally difficult to apply and potentially insecure--becomes substantially stronger when transposing bits within bit-balanced blocks, and driven with a pseudo-random sequence. Dynamic transposition combiners seem very protective of their pseudo-random sequence (a significant problem with a Vernam [27] combiner), can frustrate a statistical frequency-analysis of the ciphertext, and can guarantee strong mixing performance even with an input of unbalanced plaintext distributions.

ACKNOWLEDGMENTS

My thanks to the referees, whose questions about the utility of this mechanism led to the inclusion of material on numerical and theoretical secrecy, and to Edward Rupp for conversations about potential attacks.

REFERENCES

1. Beker, H. and F. Piper. 1982. *Cipher Systems.* New York: John Wiley & Sons.

2. Beker, H. and F. Piper. 1985. *Secure Speech Communications.* London/Orlando: Academic Press.

3. Costas, J. 1981. The Hand-Held Calculator as a Cryptographic Machine. *Cryptologia.* 5:94 - 117.

4. Costas, J. 1981. Letter to the Editor. *Cryptologia.* 5:210-212.

5. Davies, D. and W. Price. 1984. *Security for Computer Networks.* New York: John Wiley & Sons.

6. Deavours, C, D. Kahn, L. Kruh, G. Mellen, and B. Winkle. 1987. *Cryptology Yesterday, Today, and Tomorrow.* Norwood, Mass: Artech House.

7. Deavours, C. 1987. *Cryptanalytic Programs for the IBM PC.* Laguna Hills, CA: Aegean Park Press.

8. Denning, D. 1982. *Cryptography and Data Security.* Reading, Mass: Addison-Wesley.

9. Diffie, W. and M. Hellman. 1979. Privacy and Authentication: An Introduction to Cryptography. *Proceedings of the IEEE.* 67: 397-427.

10. Durstenfeld, R. 1964. Algorithm 235, Random Permutation, Procedure SHUFFLE. *Communications of the ACM.* 7: 420.

11. Feistel, H. 1973. Cryptography and Computer Privacy. *Scientific American.* 228: 15-23.

12. Gaines, H. 1956 (original 1939). *Cryptanalysis.* New York: Dover Publications.

13. Hellman, M. 1977. An Extension of the Shannon Theory Approach to Cryptography. *IEEE Transactions on Information Theory.* IT23: 289-294.

14. Kahn, D. 1967. *The Codebreakers.* New York: Macmillan.

15. Klingler, L. 1981. Letter to the Editor. *Cryptologia.* 5:209-210.

16. Knuth, D. 1981. *The Art of Computer Programming,* Vol. 2, *Seminumerical Algorithms.* 2nd ed. Reading, Mass: Addison-Wesley.

17. Kullback, S. 1976 (original 1938). *Statistical Methods in Cryptanalysis.* Laguna Hills, CA: Aegean Park Press.

18. Massey, J. 1988. An Introduction to Contemporary Cryptology. *Proceedings of the IEEE.* 76: 533-549.

19. Mellen, G. 1973. Cryptology, computers, and common sense. *Proceedings of the National Computer Conference.* 42: 569-579.

20. Mellen, G. 1983. Cryptanalysts' Corner. *Cryptologia.* 7: 371.

21. Meyer, C. and S. Matyas. 1982. *Cryptography: A New Dimension in Data Security.* New York: John Wiley & Sons.

22. Michener, J. 1985. The "Generalized Rotor" Cryptographic Operator and Some of Its Applications. *Cryptologia.* 9: 97-113.

23. Pfleeger, C. 1989. *Security in Computing.* Englewood Cliffs, New Jersey: Prentice Hall.

24. Rubin, F. 1987. Foiling An Exhaustive Key-Search Attack. *Cryptologia.* 11: 102-107.

25. Shannon, C. 1949. Communication Theory of Secrecy Systems. *Bell System Technical Journal.* 28: 656-715.

26. Sinkov, A. 1966. *Elementary Cryptanalysis.* Washington, DC: The Mathematical Association of America.

27. Vernam, G. 1926. Cipher Printing Telegraph Systems. *Transactions of the AIEE.* 45: 295-301.

BIOGRAPHICAL SKETCH

Terry Ritter is a registered Professional Engineer, a member of IEEE and ACM, with a background in computer architecture, hardware, and software. He has enjoyed spending the past few years being Blue Jean Software and Blue Jean Computer Engineering.

Terry Ritter, his current address, and his top page.

Last updated: 1995-11-11

PUBLISHED: Ritter, T. 1986. The Great CRC Mystery. *Dr. Dobb's Journal of Software Tools*. February. 11(2): 26-34, 76-83.

To read the complete article off-line, save these files: Listing One (CRCLIST1.HTM), Listing Two (CRCLIST2.HTM).

The Great CRC Mystery

Terry Ritter

The Cyclic Redundancy Check (or CRC), is a way to detect errors in data storage or transmission. With more and more data being transmitted over phone lines, the need for protocols that protect data from damage in transit as increased, but the theory behind CRC generation is not well known.

What Is a CRC?

The Cyclic Redundancy Check is a way to detecting small changes in blocks of data. Error detection is especially important when computer programs are transmitted or stored, because an error of even one bit (perhaps out of hundreds of thousands) is often sufficient to make a program faulty. Although a few errors in a text file might be acceptable (since the text can be reedited when received or recovered), an error-free file is preferable. An error-correcting protocol triggered by CRC error-detection can provide this accuracy at low cost.

The CRC algorithm operates on a block of data as a unit [1]. We can understand the CRC better if we see a block of data as a single (large) numerical value. The CRC algorithm divides this large value by a magic number (the CRC polynomial or generator polynomial), to produce a remainder, which is our CRC result.

The CRC result can be sent or stored along with the original data. When the data is received (or recovered from storage) the CRC algorithm can be reapplied, and the latest result compared to the original result. If an error has occurred, we will probably get a different CRC result. Most uses of CRC do not attempt to classify or locate the error (or errors), but simply arrange to repeat the data operation until no errors are detected.

Using the CRC

The IBM 8-inch floppy disk specification used the CRC-CCITT polynomial for error-detection, and this CRC is now used in almost all disk controller devices. A disk controller computes a CRC as it writes a disk sector, and then it appends that CRC to the data. When the data is read back, a new CRC is computed from the recovered data and compared to the original CRC. If the CRC values differ, an error has occurred and the operation is repeated. The standard disk CRC (CRC-CCITT) is hidden in the controller, and nowadays receives little comment.

One version of the XMODEM (or Christensen) file transmission protocol uses the CRC-CCITT polynomial to detect data transmission errors, typically caused by line noise. When the receiving end detects a data error, it sends a NAK (Negative Acknowledge) character to the sender, which requests that the data block be retransmitted. The receiving end repeats this process until the CRC from the transmitting end matches the local result, or until one or both ends give up. When the result does match, the receiving end sends an ACK (ACKnowledge) character, and the transmitting end then sends the next block.

Error Control and Efficiency

Many different CRC polynomials are possible; these generator polynomials are designed and constructed to have desirable error-detection properties. If the CRC polynomials are "well constructed" the major difference between them is in their length. Longer polynomials provide more assurance of data accuracy and are fully usable over larger amounts of data; however, longer polynomials also produce longer remainder values, which add additional error-checking overhead to the data.

A "16-bit" polynomial has a 16-bit remainder. There are two well-known 16-bit polynomials: CRC-16 (used in early BISYNC protocols) and CRC-CCITT (used in disk storage, SDLC, and XMODEM CRC). Of the two, CRC-CCITT may be a little stronger, and, by convention is often used in ways which strengthen its error-detection capabilities. This article illustrates CRC-CCITT, which is the polynomial $x^{16} + x^{12} + x^5 + 1$.

Polynomials are classified by their highest non-zero digit (or place) which is termed the degree of the polynomial. Both CRC-16 and CRC-CCITT are of degree 16, which means that bits 16 through 0 are significant in their description; a degree 16 polynomial has 17 bits. Normally we are most concerned with the remainder of the CRC operation, which has one bit less than the polynomial. Thus, we may think of 16-bit CRCs, even though their generator polynomials actually contain 17 bits (bits 16 through 0).

In a proper CRC polynomial, both the most significant bit (MSb) and least significant bit (LSb) are always a '1'. Because the highest bit of the polynomial is always a '1', we are able to treat this bit differently from the other bits of the polynomial. Since the remainder from a 16th degree polynomial has only 16 bits, a 16-bit register is sufficient for CRC operations on a 16-bit polynomial, even though the polynomial itself actually has 17 bits.

A well-constructed CRC polynomial over limited-size data blocks will detect any contiguous burst of errors shorter than the polynomial, any odd number of errors throughout the block, any 2 bit errors anywhere in the block, and most other cases of any possible errors anywhere in the data [2]. So every possible arrangement of 1, 2, or 3 bit errors will be detected. Nevertheless, there remains a small possibility that some errors will not be detected. This happens when the pattern of the errors results in a new value which, when divided, produces exactly the same remainder as the correct block. With a properly constructed 16-bit CRC, there is an average of one error pattern which will not be detected for every 65,535 which would be detected. That is, with CRC-CCITT, we should detect be able to detect 65535/65536ths or 99.998 percent of all possible errors.[3]

There is no technique which we can use to absolutely guarantee detection of any error; but we can minimize undetected errors at reasonable cost. Other error-detection techniques are available, such as checksum or voting, but these have poorer error-detection capabilities. For example, the single-byte checksum (used in the original version of XMODEM) appears to be about 99.29 percent accurate [4], which seems pretty good. But for a single additional byte, the CRC technique is about 460 times less likely to let an error pass undetected. In practice, the difference is much greater because the CRC will detect all cases of just a few errors, and these cases are more common. The cost is a 2-byte CRC value in every block. For example, the XMODEM protocol sends data in 128-byte blocks; these blocks can be CRC error-checked with an additional two bytes--an error-check overhead of about 1.5 percent [5].

Polynomial Arithmetic

The CRC performs its magic using polynomials modulo two arithmetic. Polynomial arithmetic mod 2 allows an efficient implementation of a form of division that is fast, easy to implement, and sufficient for the purposes of error detection. (This scheme is not particularly useful for the division of common numbers). Polynomial arithmetic mod 2 differs slightly from normal computer arithmetic, and is generally the most confusing part of the CRC.

A polynomial is a value expressed in a particular algebraic form, that of: $A[n]*X^n + A[n-1]*X^{n-1} + \ldots + A[1]*X + A[0]$ (or $A_n X^n + A_{n-1} X^{n-1} + \ldots + A_1 X + A_0$).

Our common number system is an implied polynomial of base 10: Each digit means that digit is multiplied by the associated power of 10. The base 2 or binary system of numeration is also a form of the general polynomial concept. When we see a number, we think of it as a single value; we mentally perform the polynomial evaluation in the assumed base to get a single result. On the other hand, formal polynomials are considered to be a list of multiplicative coefficients, and other operations are meaningful, and the ultimate single value for the polynomial may not be important.

Because decimal arithmetic uses constant-base polynomials, all of us already know how to do polynomial arithmetic in a constant base (10); however, the polynomials used in CRC calculations are polynomials modulo two. By modulo 2 we mean that a digit can have only values 0 and 1. Of course, this is always the case with binary values, so one might well wonder what all the mumbo-jumbo is about. The difference is this. A modulo polynomial has no carry operation between places [6]; each place is computed separately. We perform mod 2 operations logically, bit by bit; in mod 2, the addition operation is a logical exclusive-OR of the values, and mod 2 subtraction is exactly the same (exclusive-OR) operation.

Modulo arithmetic is used for CRC's because of its simplicity: Modulo arithmetic does not require carry or borrow operations. In computing hardware, the carry circuitry is a major part of arithmetic computation, and is a major contributor to speed limitations. Of course, since we have both subtraction and exclusive-OR instructions available in most computer instruction sets, this particular advantage is less important for software implementations of CRC. Nevertheless, the simplicity of modulo arithmetic allows several different software approaches not available in our conventional arithmetic. Note that the modulo-type operations available in programming languages (e.g., the Pascal MOD operator), operate on entire numbers rather than individual bits or places.

A polynomial division mod 2 is very similar to common binary division, except that we perform a logical exclusive-OR operation instead of a binary subtraction. Similarly, because "greater than" and "less than" are meaningless in modulo arithmetic, we can replace these operations by performing the exclusive-OR operation is the high bit is set or 1, dividing the high part of the dividend to zeros.

We can implement a polynomial division as follows: A polynomial division register of a length corresponding to the remainder produced by the polynomial to be used is set up (see Figure 1, below) [7]. Each element of the register should be able to hold the maximum modulo value; in "mod 2," a single bit suffices. (Note that the hardware diagrams are intended only as examples; very short CRC's are of limited practical use, and there are better ways to do the job.)

```
        Polynomial = x^5 + x^4 + x^2 + 1 = 110101

        x^4       x^3     x^2         x^1     x^0
         +---+     +---+   +---+       +---+   +---+
+<-XOR<-|Q D|<-XOR<-|Q D|<--|Q D|<-XOR<-|Q D|<--|Q D|<--|Q D|<-XOR<- dn
|  |   |   ^  |   |   |  |   ^  |   |   |  |   |   |  |   |   ^
|  +---+   |  +---+   |  +---+   |  +---+   |  +---+   |  +---+   |
|          |          |          |          |          |          |
+----------+----------+----------+----------+----------+----------+

   FIGURE 1.  Polynomial Divide Hardware for a 4 bit CRC
```

The register is cleared, then the data are shifted into the register from the right; each shift is a polynomial multiplication. Each shift also shifts a bit out of the register from the most significant bit (MSb). We know that the register value will exceed one representation when the polynomial is not in logical 1, so we arrange to perform our polynomial subtraction" when this happens; that is, when we shift out a 1, we exclusive-OR the polynomial with the value in the register. Because our polynomial (the magic number) always contains a high-order bit, which always forces the shifted-out bit back to a logical 1, we need not actually operate on the high-order bit. So only zeros shift out, keeping the mod 2 polynomial remainder in the register.

This bit-level hardware process is easily simulated. Turbo Pascal algorithms for the simulation are shown in Listing One, page 76. Software simulation has the advantage of a fast and easy investigation of an algorithm, allowing quick changes to try out various forms of optimization. The program produces a "trace" of the execution, showing the step-by-step operation.

The polynomial division register does not hold the desired remainder until the place containing the last data bit has been shifted out of the register. To do this, a zero data bit be shifted in for each bit of the register. In the case of CRC-CCITT, 16 bits (2 bytes) of zeros need to be appended to the data. After entering the zero bits, the result in the polynomial division register is the CRC result. The continuing implementation of XMODEM requires these two trailing bytes.

The CRC result can be obtained without shifting in the two zero bytes by rearranging the CRC register and feeding the data in at the top end of the system (see Figure 2, below). By shifting the CRC register we can shift zeros in from the right. The data bit will be compared to the MSb in the CRC register, and only if they differ will the polynomial be subtracted. As before, this acts to keep the full remainder in the register; however, the remainder is now correct after each bit, and requires no trailing zeros.

```
        Polynomial = x^5 + x^4 + x^2 + 1 = 110101

        dn   x^4       x^3     x^2         x^1     x^0
         V   +---+     +---+   +---+       +---+   +---+
+<-XOR<-|Q D|<-XOR<-|Q D|<--|Q D|<-XOR<-|Q D|<--|Q D|<--|Q D|<-+
|  |   |   ^  |   |   |  |   ^  |   |   |  |   |   |  |   |   |
|  +---+   |  +---+   |  +---+   |  +---+   |  +---+   |  +---+   |
|          |          |          |          |          |          |
+----------+----------+----------+----------+----------+----------+

   FIGURE 2.  CRC Hardware
```

A simulation of this immediate-result algorithm (called, for lack of a better name, the CRC algorithm) is also given in Listing One, for comparison to polynomial division. Notice that both the polynomial division and CRC algorithms come up with the same remainder (or CRC value), but the CRC version does it faster and with more consistent logic.

Faster CRC's in Software

The bit-by-bit form of the CRC algorithm can be, and often is, directly simulated in software. The shifting and looping required by this approach can be reduced in several ways. Both byte-oriented [8] and table-oriented [9] algorithms have been available in the technical literature for a number of years. Table-oriented algorithms may (or may not) produce somewhat higher speed, at the expense of a relatively-large set of constants that generally must be initialized before use. Examples of the various forms of CRC algorithms are given in Listing Two (page 78).

We can speed up the algorithm even more by precomputing the CRC for all possible combinations of a 16-bit CRC and a data byte and then saving the results. Done naively, this would be a transformation of 24 bits (16 bits of the previous register, and 8 bits of data) into 16 bits. This approach would thus require 2^25 Bytes (about 34 Megabytes) of look-up table. In order to make the table approach practical, we must find a way to reduce the size of the table.

If we examine the CRC hardware, we notice that the current data bit is always combined with the current MSb of the CRC register. When we compute a whole byte CRC, we end up combining the whole data byte with the high-byte of the CRC. We can precompute the exclusive-OR of the data byte and the high byte of the CRC register (this is a single operation in software), yielding a single byte we can call the combined term or the combination value.

For the common 16-bit CRC's, it turns out that the CRC register changes in patterns which are directly related to the combination value. Thus, it is possible to pre-compute the CRC changes for all 256 possible combination values. Then, when we need to do a CRC, we can use the 1-byte combination value to look up a corresponding 2-byte result, then use that result to correctly change the CRC register. As one might expect, the required change is simply a 2-byte exclusive-OR operation.

To generate the data for the lookup table, we need only generate the 2-byte CRC result for all 256 possible data bytes, given an "all zeros" starting CRC register. Each result has a 1 for those bits in the CRC register that are changed by a particular combination value. We can use a nonlinear implementation of the CRC to compute the table values.

This approach to generating a table of CRC values thus requires a 512-byte lookup table. We must fill the table with the correct data in an initialization step and perform a few more run-time operations than the straight lookup process requires (compute the combination value, look up the result, then apply the result to the CRC register and compute the new CRC value).

Another variations that is faster than the simple table-and-lookup approach, but also eliminates the look-up storage of the table approach is the bytewide shifting algorithm. A bytewide approach eliminates the bit-by-bit test-and-jump operations which are a significant overhead in the bit-by-bit version, and also takes advantage of fast shift and parallel-logic operations available on most processors (as well as some high-level languages such as Turbo Pascal or C).

First we need some algebra: By giving each CRC register bit and each data bit a separate symbol, we can express the result of a CRC operation symbolically. Each bit of the CRC register will be represented by a formula showing all the data and original CRC bits which affect that bit in the result. If we take the exclusive-OR of the bits specified by the formula, we can directly compute any bit of the CRC result.

In order to generate the formulas for each bit of the CRC register, we create an algebraic analog of the shifting and combining process of the bit-by-bit CRC algorithm. Instead of shifting bit values (as in a normal shift register), we instead move the whole symbolic formula for each bit to the next higher bit position. Instead of actually performing an exclusive-OR operation, we concatenate the formula for the data bit to each of the affected bits in the CRC register, with a symbol indicating an exclusive-OR operation. If ever we find that we have two identical variables in any one formula, we can cancel and eliminate them both (because anything exclusive-ORed with itself is zero, and zero exclusive-ORed with any value is just that value).

After symbolically processing a whole byte of data, and eliminating common terms, we come up with a symbolic representation for each bit of the result. By factoring this expression into convenient computer operations, a program is obtained which implements the bit-by-bit parallelism available in software.

CRC Deviations

More improvement is possible. We have previously assumed that the CRC register is cleared before starting the computation, and also that we specifically compare the stored (or transmitted) CRC value to the current CRC result. These assumptions are discarded in protocols other than XMODEM [10].

When a CRC register is contains only zeros, processing a zero data bit does not change the CRC remainder. So, if the CRC register is clear, and extraneous zero bits do occur, these data errors will not be detected. For this reason, most current CRC protocols initialize the CRC register to all 1's before they start the computation, allowing the detection of extraneous leading zeros.

We can also eliminate the need to detect the separate CRC field at the end of a block data block. If the CRC result is simply attached to the end of the data, the receiving CRC register will clear itself automatically if there is no error; that is, each bit of the stored or transmitted CRC value should cancel the similar bit in the CRC register. Although of minor importance for software implementations, this is a reasonable simplification for hardware CRC devices because it allows the exact same hardware to be used regardless of block length.

When the CRC is appended to the end of the data (thus eliminating the need to detect it as a separate field), and if bit-level CRC hardware is also to be supported, CRC software may need to use data in reverse bit order. This is because bit-level CRC hardware works on data after it has been serialized, and data is traditionally serialized LSb-first. That is, the parallel-to-serial conversion in an asynchronous serial device sends the rightmost bit of a character first and the leftmost bit last. The bit-level CRC hardware has little choice but to treat the resulting data stream as a single larger number, but that data-stream has its byte-level bit-order changed from our usual expectations.

If a MSb-leftmost CRC routine is to be compatible with bit-level CRC hardware, it may be necessary to reverse the bit order of every data byte (before each is processed or serialized) and also the CRC remainder bytes (after the block ends). Bit-order reversal can be done in software, hardware, or both. Alternately, the CRC algorithm could be constructed so as to use and hold MSb-rightmost data.

In strictly software CRC implementations, however, we work on data before it is serialized and after it is recovered and we trust any serialization that occurs to be transparent. We can thus afford to treat data as a single large value, MSb-leftmost, with MSb-leftmost bytes and a similar MSb-leftmost CRC remainder appended on the right. This arrangement is most consistent with both the theory and our conventions, and is the form used by XMODEM. The CRC routines shown in this article use MSb-leftmost data and keep the result also in MSb-leftmost format.

If we arrange to verify the CRC by processing the CRC result as data, we again fall prey to extraneous zero data bits. In order detect such errors, we arrange for the CRC register to take on a unique nonzero value in the event of no error. By some quirk of the algebra, it turns out that if we transmit the complement of the CRC result and then CRC-process that as data upon reception, the CRC register will contain a unique nonzero value depending only upon the CRC polynomial (and the occurrence of no errors). This is the scheme now used by most CRC protocols, and the magic remainder for CRC-CCITT is $1D0F_{Hex}$.

Actual CRC Implementations

I constructed several CRC implementations for speed and size comparisons (see Listing Two). The CRC-CCITT polynomial was used, since this is the polynomial used in XMODEM, as well as many other data communication uses. I used Turbo Pascal although the code could obviously be rewritten in C. A couple of the operations used are Turbo Pascal extensions: *Swap(I)* is an INTEGER function that exchange the high and low byte of an integer value; *Lo(I)* is an INTEGER function which selects only the low byte of an integer.

I used the Pascal Bit-by-Bit approach (a direct simulation of the hardware method) to provide a reference against which the other algorithms are compared. The Pascal Fast B-B-B is an improved bit form comparable to most high-level language implementations of the XMODEM CRC, except that this version requires no trailing zeros to finish the calculation (and so is already faster than the usual Pascal Bit-by-Bit version). The Pascal Byte implements the improvement from algebraic factoring; the Pascal Table version shows how a pre-computed table can simplify and speed execution-time operation. The Machine Code versions of Byte and Table show yet more improved speed. The different approaches illustrate various tradeoffs of speed, space, and specialization. The results (Table 1, left) show a range of almost two orders of magnitude in execution speed.

Each CRC implementation was made a Pascal PROCEDURE for easy testing and comparison. For validation, varying amounts of program code from main-memory were processed by each implementation. All algorithms achieved the same result. Several of these versions have been placed in an implementation of XMODEM with good results.

Time Tests

For the time tests, each implementation was executed 10,000 times under Turbo Pascal 3.01A on an 8088 in a Leading Edge PC with a 7.16 Megahertz (MHz) clock; the times would be 50% longer on an IBM PC. The time was taken automatically from MS DOS. Because the MS DOS timer ticks only about 18.2 times per second, this method is only precise within about 55 milliseconds (msec) on both the start and end of the timing interval. The large number of repetitions minimize the effect.

```
                        10,000 Uses (secs)     1 Use (msec)
                     Procedure  In Line   Procedure  In Line

Pascal Bit-by-Bit     13.790   13.070     1.379     1.307
Pascal Fast B-B-B      7.310    6.590     0.731     0.659
Pascal Byte            2.150    1.430     0.215     0.143
Pascal Table           1.430    0.710     0.143     0.071
Machine Code Byte      1.050    0.330     0.105     0.033
Machine Code Table     0.890    0.170     0.089     0.017

TABLE 1.
```

The time reported as "10000 Uses" is real time decreased by the amount of time taken by 10,000 empty loops, thus giving us the time associated with the procedure call and execution, instead of also including the looping structure that we use only for the tests. The "In-Line" column decreases "10000 uses" by the time taken for 10,000 procedure calls and returns, giving the time for execution only.

Selection Criteria

The time necessary to process a byte (including the CRC operation, and whatever queuing operations and other tests need to be performed) should be less than the time it takes to receive a character. We could just accumulate the data in a block as it is received, then CRC-process the whole block, but this would add some delay, or latency, between receiving the last data byte and returning a response to the sender (ACK for good data, NAK for an error, in XMODEM). Some XMODEM implementations appear to use this method, giving the impression that the protocol or the CRC are responsible for the delay. Because fast CRC routines are obviously possible, it is hard to rationalize any latency at all [11].

The Pascal Byte version, which takes only a few lines of code and is machine-independent (under Turbo Pascal), may be suitable for speeds up to 9600 bps, and is a reasonable choice for most use. The Pascal Table version is a little faster, but the table generally must be initialized before use, either by using a different CRC version, or perhaps by reading the values in from a file. Alternately (in most languages) the table could be defined in the source code as a large body of constants.

The faster versions can generally benefit from being used in-line (that is, not as procedures) to avoid procedure call/return overhead, but this is also inconvenient, since each use would involve modification of the same code in different places. The Machine Code Table version is shortest, and so would minimize the duplication penalty. The Pascal Table version can also be used in-line, because it takes a minimum amount of code. I use an Include file holding the Machine Code Byte version, then call the routine as a procedure; the resulting code is both small and fast.

Other Uses

Although this article has concentrated on CRC's in communications and data storage, CRC's can be used in many different applications involving error detection. Such applications include start-up verification of ROM code, load-time verification of RAM modules (as in the 6809 operating system OS9), and program and data correctness validation.

It should be noted that CRC polynomials are designed and constructed for use over data blocks of limited size; larger amounts of data will invalidate some of the expected properties (such as the guarantee of detecting any 2-bit errors). For 16-bit polynomials, the maximum designed data length is generally 2^15 - 1 bits, which is just one bit less than 4K bytes. Consequently, a 16-bit polynomial is probably not the best choice to produce a single result representing an entire file, or even to verify a single EROM device (which are now commonly 8K or more). For this reason, the OS9 polynomial is 24 bits long.

How To Learn More

A good introduction to CRC's can be found in the classic *Error Correcting Codes*, by Peterson and Weldon (Cambridge, Mass., MIT Press, 1972), but you can expect to do some serious math to understand it. A brief non-mathematical chapter on CRC error detection in data applications (with some good figures) is available in *Technical Aspects of Data Communication*, 2nd ed., by J. McNamara (Digital Equipment Corporation, Digital Press, 1982). The very brief section in *Computer Networks* by A. Tanenbaum is also fairly good.

Notes

1. The CRC does not require a fixed block size (though there is a built-in maximum), but some error-correcting protocols do. Larger amounts of data are simply partitioned into blocks that are considered separately.
2. Peterson, W. W. and D. T. Brown. 1961. Cyclic Codes for Error Detection." *Proceedings of the IRE*. January. 228-235.
3. Tanenbaum, A. 1981. *Computer Networks*. Prentice-Hall. 128-132.
4. Brooks, L. and J. Rasp. 1984. "How Accurate is Accurate?" *DDJ*. February. 27.
5. Error detection is only part of the requirements for a protocol. Other requirements include transmitting the data in blocks, numbering the blocks, and responding when a block has been received. The corresponding design decisions in XMODEM typically add yet another four bytes to each block transferred, for a required overhead of about 4.5 percent. This value can be, and often is, additionally degraded in implementation.
6. The general case of polynomial arithmetic, which allows a nonconstant base, generally makes carry operations (between terms) difficult.
7. It is common and traditional for the CRC register to be shown shifting right, which is the exact inverse of this author's analogy to binary division. Given our system of numeration, it seems reasonable to place the most significant digits of a value to the left, and it is then correct for the CRC register to be seen as shifting to the left.
8. ❍ Helness, K. 1974. "Implementation of a Parallel Cyclic Redundancy Check Generator." *Computer Design*. March. 91-96.
   ❍ Vasa, S. 1976. "Calculating an Error-Checking Character in Software." *Computer Design*. May. 190-192.
   ❍ Socha, H., *et al.* 1979. "Letter to the editor." *Computer Design*. May. 6, 12.
   ❍ Kjelberg, I. 1985. "Letter to the editor." *MICRO*. August. 4, 99.
9. ❍ Whiting, J. 1975. "An Efficient Software Method for Implementing Polynomial Error Detection Codes." *Computer Design*. March. 73-77.
   ❍ Perez, A. 1983. "Byte-wise CRC Calculations." *IEEE Micro*. June. 40-50.
   ❍ Schwaderer, D. 1985. "CRC Calculation." *PC Tech Journal*. 118-132.
10. ❍ McKee, H. 1975. "Improved CRC Technique Detects Erroneous Leading and Trailing 0's in Data Blocks." *Computer Design*. October. 102-106.
   ❍ Fortune, P. 1977. "Two-Step Procedure Improves CRC Mechanisms." *Computer Design*. November. 116-129.
11. Some protocols other than XMODEM allow subsequent blocks to be sent before a previous block is acknowledged, thus minimizing the latency problem.

Terry Ritter, his *current address*, and his *top page*.

Last updated: 1996-04-30

Normal, Chi-Square and Kolmogorov-Smirnov Statistics Functions in JavaScript

Numerical Computations for Cryptography

Computations of combinatoric and statistics functions and inverses which deliver good accuracy over a wide range of values. Accuracy tests allow the functions to be checked in any computing environment.

A *Ciphers By Ritter* Page

Terry Ritter

Last Update: 1998 June 24

Please send comments and suggestions for improvement to: ritter@io.com. You may wish to help support this work by patronizing Ritter's Crypto Bookshop.

Contents

- **Hypothesis Testing**
- **Continuous Statistic Distributions**
- **The Meaning of Statistic Distributions**
- **Normal**
- **Chi Square** -- compare binned distribution counts
- **Kolmogorov-Smirnov** -- compare distributions without using bins

Related Pages

- **Base Conversion** -- base 2..64 to decimal and decimal to base 2..64
- **Logs**
- **Powers**
- **Permutations** -- n things taken k at a time, order matters
- **Combinations** -- n things taken k at a time, order does not matter
- **Binomial** -- for success probability p and n trials, the probability of k successes
- **Bit Changes** -- bit changes from keyed invertible substitution tables or ciphers
- **Poisson** -- given mean u, the probability of k successes

---

Hypothesis Testing

One role of statistics is to develop evidence that something *is* or *is not* performing as expected. Typically we have many objects, any one of which may be selected at random and then measured. Over many samples, measured values may recur with some fixed probability, and we can construct a frequency distribution or *density function* experimentally. If we can form a theory or *hypothesis* of what that distribution *should* be, we can compare the theoretical distribution to the experimental one. And if the distributions seem to be the same, we can have some confidence that we know what is going on: We have "proven" the hypothesis of a particular distribution.

To compare distributions, we compute some sort of "goodness of fit" statistic on the samples. Now, the experimental objects will have a distribution of values, but the test statistic has its own *different* distribution. In "goodness of fit," the statistic distribution is usually what we expect if we compare any sample distribution to itself; it is thus the variation of random sampling alone. So if we take a number of experimental samples, we can compute a statistic value. Then we can calculate, from the known distribution for that statistic, how often that result should occur by chance alone, assuming the two distributions really are the same. And if we repeatedly find statistic values which are particularly unlikely, we can conclude that the distribution of experimental objects is *not* what we expected it to be.

Continuous Statistic Distributions

The normal, chi-square and Kolmogorov-Smirnov statistics are *continuous* distributions (in contrast to the binomial and Poisson *discrete* distributions). Because continuous statistics are not limited to discrete values, there is almost no probability that a particular precise value will occur. We ask, therefore, about the probability of getting a particular value *or less*, or the value *or more*.

The probability of a particular value or less is just the area under the probability "density" curve to the left of the value; this is the "left tail." These statistics are normalized so that the total probability is 1; if we have the area to the left of a value, we can easily find the amount to the right by subtracting what we have from 1. And we can find the area between any two values by finding the area to the right value, and then subtracting the area to the left value. The same statistic probability distributions can be described from either tail, but I try to uniformly use the left tail, which is also the "cumulative distribution function" or c.d.f. The c.d.f. is just the accumulated sum or integral of the density function; it gives the area to the left of the parameter value. Whether we describe a distribution from the left tail or the right tail really does not matter as long as we understand what the associated probability means.

The Meaning of Statistic Distributions

The probability distribution for a statistic is usually what we expect to see when we sample a known distribution. Over many independent trials which sample ideal data for that statistic, we expect to find that about half of the trials will have a statistic value *below* the 0.5 probability value. Similarly, about half of the trials will have a value *above* the 0.5 probability value.

"Critical" statistic values are often on the right tail, and may indicate, for example, that 95 out of 100 trials should be below some statistic value. But this means that we *expect* to get 5 trials out of 100 *above* that critical value, as a result of random sampling. And even though a 5 percent chance is not very high, it is to be expected, and might even occur on the very first experiment.

To the extent that any statistic value can occur with some (even if fantastically unlikely) random sampling, I claim that simply finding a relatively unlikely statistic value yields no conclusion in itself. Of course, a high cost of experimentation may simply demand *some* decision which is more often right than wrong. We rarely have that situation in cryptography. But if we *repeatedly* get many more than 5 percent of the trials in the upper 5 percent of the distribution area, this is statistical *evidence* that the distribution is not what we thought it was.

In addition to worrying about the upper 5 percent or 1 percent of the distribution, we *also* expect to see about a quarter of the samples in each quarter of the distribution. It can be important check this, because a valid experimental procedure *must* behave this way. If we concentrate solely on statistic values which may be too far out on the right tail (so-called "failures"), we could miss very serious problems in the experiment itself. If we repeatedly do not find about a quarter of the samples in each quarter of the distribution, we *know* the distribution is not what we expected it to be, *even if* we have very few "failures." Any deviation from what we expect means that we do not know what is going on.

A big part of the worth of statistics occurs when we can analyze a problem, predict the results, then have measurements agree with our predictions. Repeated agreement between theory and practice means that a lot of analysis and a lot of measurement instrumentation must be working properly. But if the results do *not* agree, we have only proven that we have not described what is really happening. The difference could be the result of any one of many issues in the analysis, the design of the experiment, or its conduct. So in "goodness-of-fit" testing, we only get "proof" when the two distributions are measurably the same, and *no proof at all* when they differ.

Normal

The normal distribution is the "bell shaped" curve we associate with grading curves and other population models. The normal distribution -- or something close to it -- appears often in nature. This probably happens because the sum of many "independent identically distributed" random variables has a normal distribution, independent of the distribution of the variables taken one-by-one.

The normal is actually a whole family of curves, which differ in their mean value and in the square root of their variance or standard deviation. Here the user can enter either the variance or standard deviation value, and the other will be updated automatically. The standardized result appears in z.

Convert to Standard Z

mean [ ] variance [ ] std. dev. [ ]

y [ ]

The Cumulative Normal

z [ ]

cdf(z) = P [ ] Q = 1-P [ ]

Values of Z for Various P

0.01 [ ]	0.05 [ ]	0.10 [ ]
0.25 [ ]	0.50 [ ]	0.75 [ ]
0.90 [ ]	0.95 [ ]	0.99 [ ]

The Inverse Normal

P or Q [ ]

z [ ]

Chi Square

Chi-square is the best known goodness of fit statistic. Chi-square is a way to numerically compare two sampled distributions:

1. Some number of "bins" is selected, each typically covering a different but similar range of values.
2. Some much larger number of independent observations are taken. Each is measured and classified in some bin, and a count for that bin is incremented.
3. Each resulting bin count is compared to an expected value for that bin, based on the expected distribution. Each expectation is subtracted from the corresponding bin count, the difference is squared, then divided by the expectation:

$$X^2 = \text{SUM}(\ (\text{Observed}[i] - \text{Expected}[i])^2 \ / \ \text{Expected}[i]\ )$$
$$i$$

The sum of all the squared normalized differences is the chi-square statistic, and the distribution depends upon the number of bins through the *degrees of freedom* or *df*. The df value is normally one less than the number of bins (though this will vary with different test structures). Ideally, we choose the number of bins and the number of samples to get at least ten counts in each bin. For distributions which trail off, it may be necessary to collect the counts (and the expectations) for some number of adjacent bins.

The chi-square c.d.f. tells us how often a particular value or lower would be seen when sampling the expected distribution. Ideally we expect to see chi-square values on the same order as the df value, but often we see huge values for which there really is little point in evaluating a precise probability.

The Cumulative Chi-Square

df [ ] X^2 = x [ ]

cdf(df,x) = P [ ] Q = 1-P [ ]

Values of X^2 for Various P

0.01 [ ]	0.05 [ ]	0.10 [ ]
0.25 [ ]	0.50 [ ]	0.75 [ ]
0.90 [ ]	0.95 [ ]	0.99 [ ]

The Inverse Chi-Square

P or Q [ ]

x [ ]

Kolmogorov-Smirnov

Kolmogorov-Smirnov is another goodness of fit test for comparing two distributions. Here the measurements need not be collected into "bins," but are instead re-arranged and placed in order of magnitude:

1. *n* independent samples are collected and arranged in numerical order in array *X* as *x*[0]..*x*[*n*-1].
2. *S*[*x*[*j*]] is the cumulative distribution of the sample: the fraction of the *n* observations which are less than or equal to *x*[*j*]. In the ordered array this is just (*j*+1)/*n*).
3. *F*(*x*) is the reference cumulative distribution, the probability that a random value will be less than or equal to *x*. Here we want *F*(*x*[*j*]), the fraction of the distribution to the left of *x*[*j*] which is a value from the array.

There are actually at least three different K-S statistics, and two different distributions:

The "one-sided" statistics are:

```
Dn+ = MAX( S(x[j]) - F(x[j]) )
    = MAX( ((j+1)/n) - F(x[j]) )

Dn- = MAX( F(x[j]) - S(x[j]) )
    = MAX( F(x[j]) - (j/n) )
```

where "MAX" is computed over all *j* from 0 to *n*-1. Both of these statistics have the same distribution.

The "two-sided" K-S statistic is:

```
Dn* = MAX( ABS( S(x[j]) - F(x[j]) ) )
    = MAX( Dn+, Dn- )
```

and this has a somewhat *different* distribution.

Knuth II multiplies Dn+, Dn- and Dn* by SQRT(n) and calls them Kn+, Kn- and Kn*, so we might say that there are at least six different K-S statistics. So what do we use?

It turns out that the Dn* distribution is hard to compute with accuracy over the full range. There is a good transformation between the two distributions for values well out on the right tail, but this means we lose values for quarters of the distribution, and this is a significant loss. We also lose the ability to compute an arbitrary inverse. So, just like Knuth II, we support the "one-sided" versions.

The c.d.f. computation should give 4 good decimal places. The critical value and inverse computations iterate the c.d.f. with bisection for *n* under 100, which gives full accuracy, but is very, very slow. The critical values might take half a minute to come up, and the tests might well take a couple of minutes.

Select Kolmogorov-Smirnov Scaling

Dn+ and Dn- ○
Kn+ and Kn- ○

The Cumulative Kolmogorov-Smirnov

n [ ] Dn or Kn = x [ ]

cdf(n,x) = P [ ] Q = 1-P [ ]

Values of Dn or Kn for Various P

0.01 [ ]	0.05 [ ]	0.10 [ ]
0.25 [ ]	0.50 [ ]	0.75 [ ]
0.90 [ ]	0.95 [ ]	0.99 [ ]

The Inverse Kolmogorov-Smirnov

P or Q [ ]

x [ ]

Terry Ritter, his *current address*, and his *top page*.

Base Conversion, Logs, Powers, Factorials, Permutations and Combinations in JavaScript

Numerical Computations for Cryptography

Computations of combinatoric and statistics functions and inverses which deliver good accuracy over a wide range of values. Accuracy tests allow the functions to be checked in any computing environment.

A *Ciphers By Ritter* Page

Terry Ritter

Last Update: 1998 June 24

Please send comments and suggestions for improvement to: ritter@io.com. You may wish to help support this work by patronizing Ritter's Crypto Bookshop.

Contents

- **Base Conversion**
- **Logs**
- **Powers**
- **Factorials**
- **Permutations** -- n things taken k at a time, order matters
- **Combinations** -- n things taken k at a time, order does not matter

Related Pages

- **Binomial** -- for success probability p and n trials, the probability of k successes
- **Bit Changes** -- bit changes from keyed invertible substitution tables or ciphers
- **Poisson** -- given mean u, the probability of k successes
- **Normal**
- **Chi Square** -- compare binned distribution counts
- **Kolmogorov-Smirnov** -- compare distributions without using bins

---

Base Conversion

Numeric values are usually represented as a sequence of digits, each of which implies some amount of a base to a particular power. We thus interpret "256" in "decimal" (base 10)as being two *hundreds* (base 10 to the second power) plus five *tens* (base 10 to the first power) plus six *ones*. But that value also can be seen as five 49's plus one 7 plus four 1's, or "514" in base 7. This is just a different way to *represent* the exact same value.

The base conversion routines supplied here support base values from 2 to 64. For bases above 10 it is necessary to use more than our usual 10 numeric digits, and it is convenient to use alphabetic letters. We can adjust the alphabet as desired.

- **To convert from a strange base to decimal**, enter the number in the **Input Value** field, and its base in the **Base** field above it, then click the "**To Dec**" button.
- **To convert from decimal into another base**, enter the number in the **Decimal Value** field and the desired **Base** above the **Output Value** field, then click the "**From Dec**" button.

Input Value Base

Input Alphabet

Decimal Value

Output Alphabet

Output Value Base

Logs

x

ln(x) $\log_2(x)$

Powers

x n

x^n ln(x^n) $\log_2(x^n)$

Factorials

The factorial of *n* is the product of all integers from *n* down to 2. While any scientific calculator will deliver a result for a small *n*, I have often needed values for large *n* beyond any calculator I have seen. And while such values may be too large for convenient representation, their base-2 logarithm tells us the number of bits needed to express the value in binary. Cipher keyspace is also expressed in bits.

n

n! ln(n!) $\log_2(n!)$

Permutations

A permutation is an *ordering*, a re-arrangement of symbols or objects. Here we calculate the number of arrangements possible given *n* symbols, with *k* of these being used in each arrangement. Again we expect to see some large values, and so also report results as base-2 logs or bits.

n k

P(n,k) ln P(n,k) $\log_2$ P(n,k)

Combinations

A combination is a particular set of symbols, independent of their ordering or arrangment. Here we calculate the number of combinations of *n* things taken *k* at a time. Again we also report results as base-2 logs or bits.

n k

C(n,k) ln C(n,k) $\log_2$ C(n,k)

Terry Ritter, his current address, and his top page.

Binomial and Poisson Statistics Functions in JavaScript

Numerical Computations for Cryptography

Computations of combinatoric and statistics functions and inverses which deliver good accuracy over a wide range of values. Accuracy tests allow the functions to be checked in any computing environment.

A *Ciphers By Ritter* Page

Terry Ritter

Last Update: 1998 June 24

Please send comments and suggestions for improvement to: ritter@io.com. You may wish to help support this work by patronizing Ritter's Crypto Bookshop.

Contents

- **Binomial** -- for success probability p and n trials, the probability of k successes
- **Bit Changes** -- bit changes from keyed invertible substitution tables or ciphers
- **Poisson** -- given mean u, the probability of k successes

Related Pages

- **Base Conversion**
- **Logs**
- **Powers**
- **Factorials**
- **Permutations** -- n things taken k at a time, order matters
- **Combinations** -- n things taken k at a time, order does not matter
- **Normal**
- **Chi Square** -- compare binned distribution counts
- **Kolmogorov-Smirnov** -- compare distributions without using bins

---

Binomial

The binomial distribution represents the probability that a particular type of event will occur a given number of times. When each "success" has probability *p*, and there are *n* trials, we can compute the probability of getting exactly *k* successes. We can also compute the cumulative probability of getting *k* or fewer successes; this is the *cumulative distribution function* or c.d.f.

For the binomial to be an appropriate model, it must describe *Bernoulli trials*. This is sampling with replacement in which:

- Each trial is independent,
- Each outcome is random, and
- The probability of success is constant.

Since it is common in cryptography to create random-like events, these conditions are often met, although the values involved may be far larger than could be easily handled with a normal calculator.

The binomial distribution gives the probability of finding exactly ***k* successes** (or *at least k* successes in the c.d.f.) over ***n* trials** given **probability *p* of success** on each trial:

```
                n     k       n-k
   B(k,n,p) = (  )   p   (1-p)
                k
```

p [ ] **n** [ ] **k** [ ]

Exactly k [ ]

cdf = P(0..k) [ ] **Q = 1-P** [ ]

Bit Changes

The binomial is the appropriate distribution for bit-changes from an invertible substitution table or cipher. This is the source of bit diffusion within a substitution table, and is related to avalanche.

Suppose we have some input value to a substitution table or cipher. Now suppose we change that input value. This will select a different value from the table. We can expect about half of the output bits to *change* on average, but we could get as few as one, or as many as all of them. The distribution is binomial.

table width, bits [ ] **samples** [ ]

BIT CHANGES

Poisson

The Poisson distribution is a simplified model for the binomial distribution under certain conditions:

- The number of events *n* is large,
- The probability of success *p* is small, and
- The expectation *np* or *u* is moderate.

The Poisson distribution gives the probability of finding exactly ***k* successes** (or *at least k* successes in the c.d.f.) given success **expectation *u***:

```
                k  -u
   P(k,u) = u  e    /  k!
```

where *e* is the base of natural logarithms:

```
   e = 2.71828...
```

and *u* is:

```
   u = n p
```

u [ ] **k** [ ]

Exactly k [ ]

cdf = P(0..k) [ ] **Q = 1-P** [ ]

Terry Ritter, his current address, and his top page.

Active Boolean Function Nonlinearity Measurement in JavaScript

A detailed discussion of cryptographic nonlinearity, what it means and how it is computed, with active JavaScript panels to perform the computation.

A **Cipher By Ritter** Page

Terry Ritter

Nonlinearity is the number of bits bits which must change in the truth table of a Boolean function to reach the closest affine function. If we believe that cryptosystems based on linear or affine functions are inherently weak, the ability to measure nonlinearity is the ability to measure one form of strength.

Nonlinearity measurement is particularly useful to quantify the strength of invertible substitution tables. This is important when pre-defined tables are a part of a cipher definition. But nonlinearity measurement can be even more important in the context of scalable ciphers. When ciphers can be shown to experimental size, it becomes possible to talk about the overall nonlinearity (for each key) of the cipher and so forth. This is the more information then we usually have on cipher design.

Affine Boolean Functions

A Boolean function produces a single-bit result for each possible combination of values from perhaps many Boolean variables. The Boolean field consists of the values {0,1}, with XOR as "addition" and AND as "multiplication."

An affine Boolean function has the form:

```
f = A0⊕A1 • A1₂₁₂₁ + ... + A1₂₁₂₁ • A0
```

In the Boolean field, a constant or **A0** value of '1' inverts or reverses the result, while a constant of '0' has no effect. The coefficients **a**, simply *enable or disable* the associated variable **x**. And if we consider the collected coefficients to be a counting binary value, we have a unique ordering for affine Boolean functions:

```
Affine Boolean Functions

   f0 = 0*x[2] + 0*x[1] + 0 = 0
   f1 = 0*x[2] + 0*x[1] + 1 = 1
   f2 = 0*x[2] + 1*x[1] + 0 = x[1]
   f3 = 0*x[2] + 1*x[1] + 1 = x[1] + 1
   f4 = 1*x[2] + 0*x[1] + 0 = x[2]
   f5 = 1*x[2] + 0*x[1] + 1 = x[2] + 1
   f6 = 1*x[2] + 1*x[1] + 0 = x[2] + x[1]
   f7 = 1*x[2] + 1*x[1] + 1 = x[2] + x[1] + 1
```

In this way, we can write 16 different forms for 3-variables. But it is convenient to just the functions which are the same except for the value of the constant, and then we have exactly 8 affine Boolean functions of 3-variables. Each of these has a particular value for every possible combination of variable value, which we can show in a truth table:

```
  Distance to an Affine Boolean Functions

  affine    truth table

                 f   f   f   f   f   f   f   f
                 0   1   2   3   4   5   6   7
  x[0]            0   1   0   1   0   1   0   1
  x[1]            0   0   1   1   0   0   1   1
  x[2]            0   0   0   0   1   1   1   1
  x[0]+x[1]       0   1   1   0   1   0   0   1
  x[0]+x[2]       0   1   0   1   1   0   1   0
  x[1]+x[2]       0   0   1   1   1   1   0   0
  x[0]+x[1]+x[2]  0   1   1   0   1   0   0   1
```

Unexpected Distance

One way to measure a sort of "correlation" between two Boolean functions is to compare their truth tables and count the number of bits which differ; this is their Hamming distance.

Since we expect about half the bit positions to differ (on average), we can subtract that expected distance and come up with what I am calling – for lack of a better term – the "unexpected distance" for the truth table. The larger the value of this new unexpected distance is, while the sign indicates the direction. Consider two functions and their difference:

```
  Unexpected Distance to Affine Boolean Function

  affine    truth table                 distance    ud

                 f   f   f   f   f   f   f   f
                 0   1   2   3   4   5   6   7
  x[0]            0   1   0   1   0   1   0   1        4        0
  x[1]            0   0   1   1   0   0   1   1        4        0
  x[2]            0   0   0   0   1   1   1   1        4        0
  x[0]+x[1]       0   1   1   0   1   0   0   1        4       +2
  x[0]+x[2]       0   1   0   1   1   0   1   0        4        0
  x[1]+x[2]       0   0   1   1   1   1   0   0        4        0
  x[0]+x[1]       0   1   1   0   1   0   0   1        4       +2
  x[0]+x[2]       0   1   0   1   0   1   0   1        4       -2
  x[1]+x[2]       0   0   1   1   0   0   1   1        4       -2
```

NEW
deja
.com
FOR YOU
Welcome! We've made our site even better! Discover these great new Deja.com features:
Get pure Usenet! Our discussion service is now an independent site. Follow the link at the top of the Deja.com homepage to get to clean and superfast Usenet access.
Shop smarter! In our Computing & Tech section (other categories coming soon), you'll now find detailed product info, powerful comparison tools and a unique pricing/availability shopping service.
Surf faster! Enjoy quicker page downloads and faster access to the information you want.
> Come on in!

SEARCH FOR PRODUCTS

in all channels | SEARCH>>

Looking for Usenet? SEARCH DISCUSSIONS

Wednesday, Apr. 05

Welcome!
Login | Register

- Arts & Entertainment
- Automotive
- Computing
- Consumer Electronics
- Health & Fitness
- Home & Family
- Lifestyles
- Money
- Politics & Media
- Recreation
- Sports
- Travel

Help User Tour
Contact Us

WHAT'S NEW

You can now check out books, CDs and videos and DVDs for price / availability and side-by-side comparisons on Deja.com's Arts & Entertainment channel.

TODAY'S TOP PRODUCTS

Compaq Aero 2100

Iomega Zip Drive

Tom Petty and the Heartbreakers: *Echo*

Lexmark Z11

START SHOPPING

Desktop PCs
When the chips are down, the only thing that will help is a new desktop PC that you've shopped smart for at Deja.com.
> Top-Rated > Product Wizard > Compare Side by Side

COMPARE

Monitors: ViewSonic vs. KDS
> Compare Side by Side > Top-Rated

ViewSonic E2 Series E771
Deja.com rating: 4.1
Best price: $219
Ranked: Top 5 percent

KDS Visual Sensations VS-195
Deja.com rating: 3.8
Best price: $279
Ranked: Top 10 percent

BUY

Sci-Fi Movies
Want to take a trip but gas costs too much? Try a sci-fi flick instead. This is where to find it at the best price.
Blade Runner Price/Availability
12 Monkeys Price/Availability
Contact Price/Availability
> **Check out all the sci-fi movies**

RATE

A Mouse Tale
Go wireless with a Logitech Cordless MouseMan Wheel. R. Kamerling, who's from Enschede, in the Netherlands, gave it an overall rating of 4.0 and says, "It's very practical to use a wireless mouse (I prefer to call it guinea pig, since this mouse lost its tail)."

> Read all the reviews of the Logitech Cordless MouseMan Wheel
> See the top-rated mice, trackballs and tablets

RATE IT NOW
- Logitech Cordless MouseMan Wheel
- Kensington Expert Mouse PC
- Microsoft Intellimouse Explorer
- Cirque Power Cat

Deja.com Special Features
- **Best of the Web** Cast a vote for your favorite Web sites in a special area developed with *Forbes* magazine.

Books, Videos, Music!
SHOP AMAZON.COM NOW

Manage Your Health
Visit CBSHealthwatch Today

Visit the Deja.com Career Center

Computer & Tech Jobs
Upgrade your Career!

Slash Prices...
In a Friendly Sorta Way

FREE Arc Radio!
When you try Netmarket.

Buy Today's Hot Sellers at **firstsource.com**
1. 3Com Palm V
2. Sony Spressa USB CD-RW
3. SimpleTech 128MB PC-100 SDRAM
4. Windows Pro 2000 Upgrade
5. Norton Internet Security 2000 v1.0

Copyright © 1995-2000 Deja.com, Inc. All rights reserved.
Trademarks · Terms & Conditions of Use · Site Privacy Statement.

Fixing Strength Problems in Cipher Use

A *Ciphers By Ritter* Page

A sci.crypt conversation on fundamental problems in cryptography.

Introduction and Comments by Terry Ritter

There is no guarantee that any cipher will not fail, but we can use ciphers in ways which **require multiple ciphers to fail** before information is exposed.

We cannot know when our cipher has been broken, but we can use ciphers in ways which **terminate any existing break** and so protect our future messages.

We cannot know the capabilities of our opponents, but we can use ciphers in ways which **force our opponents to make huge, continuing investments**, just to keep up with us.

Introduction

In several ways, cryptography can be seen as a contest: First and most importantly, it is a contest between cryptographers and unknown opposing cryptanalysts. Next, it is a contest between "our" academic cryptanalysts and the unknown cryptanalysts. Then it is a contest between independent cryptographers and academic cryptanalysts. And, of course, it is also a contest between the academic cryptanalysts.

Seeing cryptography as contest can help clarify the uneven nature of these events.

- Cryptographers have no idea who their opponents are or what capabilities they may possess (except that they will be at least at the level of the current academic literature). Cryptographers also have no way to produce "unbreakable" ciphers, or to know the ultimate "strength" of a cipher with respect to unknown attacks. Since we do not know what our ciphers can withstand, nor what capabilities they confront, we have no way to judge the probability of cipher failure. We cannot even state that failure is "unlikely."
- Currently, cryptography depends upon academic cryptanalysis to provide a form of "validation." Cryptanalysis does provide an "upper bound" on strength: the effort involved in the current best-known attack. But our opponents are not interested in paying that expense, and they seek other ways to break our ciphers. We cannot rule out such attacks, because cryptanalysis does not tell us that any minimum amount of effort at all is required to break a cipher. Cryptanalysis does not provide a "lower bound" on cipher strength, no matter how much analysis that cipher may have had.
- Our opponents work in secrecy and presumably try very hard indeed to keep their successes secret. So even if our cipher is broken, we will not know that, and so will continue to use the broken cipher. Indeed, everyone else will also continue to use that same broken cipher, unless and until an academic cryptanalyst finds a similar attack. To the extent that our academic cryptanalysts do not find a weakness the opponents do find, they unknowingly lose face -- but we continue to use a weak cipher.
- Independent cryptographers present their cipher designs which academics occasionally bestir themselves to consider. The strongest designs do not produce academic results and so are not discussed in the literature. Designs which do have problems produce results which can be published in the name of academic fame. The literature results thus naturally show an inherent bias against independent designs, and in any case are not oriented toward improving those designs.

Trouble in Paradise

It has long been known that cryptography has very serious fundamental problems. Surely we all know that no cipher system is proven secure, yet we often think we can trust a cipher that has been around for a while. These positions are contradictory, and are examples of the "dirty little secrets" which most professionals know and ignore because it seems little can be done about them. Serious issues in cryptography include:

1. **We do not know the "strength" of our ciphers**
   ("Strength" is the minimum effort needed to expose protected data). Cryptanalysis does not testify to cipher strength unless it finds a "break," and then it only gives a upper bound. The lower bound is always zero: **Any cipher can fail at any time.**
   - **We cannot "trust" any cipher**
     Any trust we may have in a cipher is more self-delusion than science, based on our not being specifically told that our cipher failed. We interpret a lack of information as evidence of strength, and this is a serious error.
   - **Older "seasoned" ciphers are not necessarily better**
     Older, more mature ciphers are generally more trusted. They also have had more time to be attacked and broken in secret.
   - **Any cipher may already be broken**
     As long as our opponents do not blab that they have broken our cipher, we will probably continue to use it.
   - **Having just a single cipher can lead to loss of all past and future data**
     Any cipher can fail at any time.
2. **We cannot know the capabilities of the opposing cryptanalysts**
   If our opponents read the open literature they are at least as advanced as "our guys." But they often work for organizations which do their own research and do not share their results. Accordingly, the academic literature does not provide a rational basis for estimating the capabilities of our opponents.
   - **As long as we continue to use the same cipher, we have no ability to terminate our opponents' success after they break that cipher**
     And we will not be told when they do succeed.
3. **We cannot estimate the chance that our ciphers may fail**
   Our opponents work in secret and do not announce their successes. We know neither the number of attempted attacks, nor the number of their successes: We thus have no basis for constructing a probability of opponent success.
   - **We cannot estimate failure from academic literature**
     Negative academic results are generally not published: We do not know the number of attempted attacks. We also cannot extrapolate academic attacks to those mounted in secret by opponents with unknown resources.
   - **Long use is not an indication of success**
     We will not know when a cipher is broken for real. Ciphers in use for a long time may have been broken and ineffective for a long time.

Fix Proposals

The conversations archived here start from my proposals to "fix" some of these serious problems. They ran into a great deal of debate and dispute, apparently because these ideas run counter to the currently accepted philosophy of cryptography. Briefly, that philosophy is to have a few ciphers which are analyzed as intensely as possible; if they survive some unknown amount of time or effort, they then transition into use. And while there is a general acceptance of the idea that even well-analyzed ciphers *may* fail, there is a remarkable lack of protocols intended to prevent such failure, or even reduce the consequences. It appears that most professionals *believe* in the ciphers they use, which may be more a testimony of the intimidating role of academic cryptanalysis than of any factual basis for such belief.

Quick Entries into the Discussion

Fix Proposals

- 1999-04-02 Terry Ritter: the first fix proposal here
- 1999-04-18 Terry Ritter: a later summary

- 1999-04-19 Terry Ritter:
- 1999-04-19 Trevor Jackson, III:
- 1999-04-20 Jim Gillogly:
- 1999-04-21 Terry Ritter: including some description of how multi-ciphering would work

Note that there are repeated descriptions and arguments for this "fix package" throughout the discussion, but they do tend to be piece-by-piece rather than as a collected unit.

The Kerkhoff's Law Delusion

Kerhkoff's laws were intended to formalize the real situation of ciphers in the field. Basically, the more we use any particular cipher system, the more likely it is that it will "escape" into enemy hands. So we start out *assuming* that our opponents know "all the details" of the cipher system, **except the key**.

Part of the fix proposal is to have many different ciphers, and to select among them "at random" **by a key**. The particular cipher used is thus unknown to our opponents; they do know, of course, that our system uses many ciphers. And if they can simply try each possible cipher, there is not much protection (although, presumably, they could not break every cipher). But with a substantial number of ciphers (say, 1000), used in a 3-level "cascade" of multi-ciphering, we have over 10**9 different cipher possibilities. And each of these is stronger than any one cipher alone, because known plaintext and defined plaintext attacks are not possible on the individual component ciphers.

The first effect we see here is keyspace, and we already have plenty of keyspace. But the effect we really want is the partitioning and compartmentalization of our data so that a failure in one cipher means the exposure of only a small part of the data.

- 1999-04-09 Andrew Haley
- 1999-04-09 Patrick Juola

The Cipher Strength Delusion

It is well-known and universally accepted that any cipher *might* be weak. When ciphers have been extensively analyzed, this possibility has generally been thought unlikely. But examination of what we know and what we cannot know shows that we have no basis for estimating such a possibility: We thus have no grounds for assuming cipher failure to be "unlikely." The immediate consequence of such a realization is that we need to do something about this in real systems.

- 1999-04-16 Terry Ritter:
- 1999-04-21 Terry Ritter:

The Probability or Risk of Cipher Failure Delusion

- 1999-04-16 Patrick Juola:
- 1999-04-19 Terry Ritter:
- 1999-04-17 Terry Ritter:
- 1999-04-21 Terry Ritter:
- 1999-04-21 John Savard:
- 1999-04-21 Terry Ritter:
- 1999-04-21 Terry Ritter:

The Confidence and Trust Delusion

- 1999-04-20 Terry Ritter:

The "No Other Choice" Delusion

- 1999-04-18 H. Ellenberger:

The "Multi-Ciphering is Bad" Delusion

- 1999-04-26 Jerry Coffin:

The "It's Not What You Say, It's Who Your Are" Delusion

- 1999-04-16 Steven Alexander:
- 1999-04-17 Trevor Jackson, III:
- 1999-04-20 Terry Ritter:

Contents

- 1999-04-02 Terry Ritter: "One of the facts of ciphering life is that we cannot prove the strength of any cipher. Even NIST review and group-cryptanalysis does not give us proven strength in a cipher, so any cipher we select might be already broken, and we would not know. We cannot change this, but we can greatly improve our odds as a user, by multi-ciphering under different ciphers."
- 1999-04-02 Boris Kazak: "And how about a 'variable' cipher?"
- 1999-04-04 Lincoln Yeoh: "I'd go further- by having layers of many and not just 3 layers of encryption, and each layer not leaking any headers at all, the attacker's work will be very much harder, even if each layer is just 40 bit crypto."
- 1999-04-09 Andrew Haley: "Kerckhoff's maxim says that your attacker knows the cryptosystem you're using, but does not know the key. If you're using superencryption, your attacker knows which systems you're using."
- 1999-04-09 Terry Ritter: ". . . if the ciphers are dynamically selected by keying, or just dynamically selected frequently by communications under cipher, the attacker does *not* know "which systems you're using." Kerckhoff's maxim does not apply."
- 1999-04-09 Patrick Juola: "If you are taking a set of cyphers and reordering them, Kerchoff's maxim suggests that you have to assume that the attacker knows the set of cyphers and just doesn't know the order."
- 1999-04-09 Boris Kazak: "It all depends on the numbers in question."
- 1999-04-09 Terry Ritter: "Every cipher is only 'known' to the Opponents after they have identified it, acquired it, analyzed it, and, presumably, broken it. All this necessarily takes time, and this time works for the user and against the attacker."
- 1999-04-9 Harvey Rook: "By Kerchhoff's maxim, you have to assume your attacker has a copy of your deciphering machine. If he has a copy of your deciphering machine, the attacker can figure out the algorithm you use to select ciphers."
- 1999-04-09 Terry Ritter: "That is no more true than saying that the attacker can figure out the message key or the initialization vector. We assume the system has some reliable random source to make such values. Similar values select what ciphers to use in what order. And this should change frequently."
- 1999-04-9 Harvey Rook: "The reliable random source must be communicated to both the encryptor, and the decryptor. Because it's transmitted or shared, you must assume the attacker has intercepted it. Because the attacker has intercepted the it, the attacker knows what ciphers you are using." "Which would you rather trust, one well analyzed cipher with 140 bits of key, or 4096 ciphers that probably aren't well analyzed, 128 bits of key, and some mechanism to flip between the ciphers."
- 1999-04-10 Terry Ritter: "That is false. It is always necessary to transport keys in some way. But whatever way this is done -- by courier, multiple channels, or authenticated public key -- random message key value are known by both ends. That keying value can be used to select ciphers." "I take the 4096 ciphers, and I have explained why: First, the multi-cipher situation forces any Opponent whatsoever to keep up in a process which is vastly more expensive for them then for us. Next, we divide our total traffic among many different ciphers. So even if an Opponent breaks a cipher, the best they can hope for is 1/n of the traffic. And in the multi-ciphering case, they don't even get that."

- 1999-04-12 ssimpson@hertreg.ac.uk: "Make the cipher selection(s) dependant on additional key material?"
- 1999-04-09 Boris Kazak: ". . . what if we would initialize BLOWFISH not with the digits of 'pi', but with some other constant - e, sqrt(2), sin(1)...? How many different ciphers can we generate from the same base algorithm . . . ."
- 1999-04-12 ssimpson@hertreg.ac.uk: "Why not make cipher selection for the compound cipher part of the key?"
- 1999-04-12 Jerry Coffin: "If you're going to include code for a number of forms of encryption, from a viewpoint of security, you might as well just always use ALL the forms of encryption supported . . . ."
- 1999-04-22 Matthew Skala: "Speed."
- 1999-04-16 Sundial Services: "When I look at most publicly-available cryptographic algorithms, I see that nearly all of them consist of round upon round of simple operations . . . ."
- 1999-04-16 John Savard: ". . . an academic researcher isn't going to take time studying a cipher that is so big and complicated that there is no hope of coming away with an impressive result . . . ." ". . . what gives real confidence in a cipher design is that it has been studied by experts who have failed to crack it . . . ."
- 1999-04-16 Terry Ritter: "The truth is that we *never* know the 'real' strength of a cipher." "Any cipher can fail at any time."
- 1999-04-16 Steven Alexander: "If I(a nobody) create a new cryptosystem tommorrow, nobody will have any confidence in it." "Only after many people have analyzed and failed to break my cipher will people say..."his cipher has held up to five(ten) years of cryptanalysis by very knowledgeable cryptanalysts. We can assume with an adequate level of confidence that the cipher will protect our information.'"
- 1999-04-16 Terry Ritter: "The issue is not who makes a thing, but instead what the thing actually is." ". . . without a LOWER bound on strength, academics REALLY do not even know that ciphers work *at* *all*, let alone how." "Unfortunately, when we use a cipher, we are very rarely concerned whether academics can break our cipher or not."
- 1999-04-16 Steven Alexander: "We can however put more trust into an algorithm that has undergone more cryptanalysis and has been tested against the newest cryptanalytic techniques because we know what will not break the cipher."
- 1999-04-16 Terry Ritter: "Nope. Simply because 'we' cannot break it does not mean that others cannot break it."
- 1999-04-16 Steven Alexander: "What exactly is your suggestion for the creation of a cipher in which we can place our trust?"
- 1999-04-17 Terry Ritter: "Absent a theory or overall test of strength, there can be no trust in a cipher. All the trust one can have is delusion." ". . . the real issue is that while supposedly everyone knows that any cipher can be weak, there has been essentially no attention given to protocols which deal with this problem."
- 1999-04-18 John Savard: "A cipher should be designed conservatively . . . in the sense of having extra complexities in its design *far beyond* those needed . . . to frustrate *known* methods of attack."
- 1999-04-18 wtshaw: "A good trick is to telescope complexities into new primatives if you can."
- 1999-04-17 ybizmt: "Refresh my memory. What do you sell?"
- 1999-04-17 Terry Ritter: "Just the truth, lately."
- 1999-04-18 John Savard: ". . . choosing a cipher from a respected source is not 'superstition', and it is the kind of choice people make all the time: i.e., when shopping for a new computer."
- 1999-04-20 Terry Ritter: ". . . this situation is not a technical discussion between people of expertise but, rather, ordinary users who really have no choice but to rely upon promotion and rumor."
- 1999-04-20 wtshaw: "I wonder if the FTC has a role in determining if claims are reasonable. They would have to yield to NSA for expertise? Perhaps we can try to shift burden directly to government to prove strength, therefore making them show their hand."
- 1999-04-17 wtshaw: "What matters is whether a cipher is good, and it will be so regardless of confidence bestowed by some select group fixated on a remarkedly few, perhaps some wrong, design criteria."
- 1999-04-17 Trevor Jackson, III: "There's a name for this attitude. It's called the Aristotelean Fallacy -- the appeal to authority."
- 1999-04-18 wtshaw: "Maturation is not the enemy of creative, but wheeler-dealer, power-sponges, who imagine that everyone else must follow their lead, are."
- 1999-04-21 Leonard R. Budney: "When rigorous proof is not available, then the opinion of an expert constitutes the best information to be had."
- 1999-04-22 Sam Simpson: "Actually, certain instances of RSA cannot be equivalent to the underlying IFP (D.Boneh, R.Venkatesan, 'Breaking RSA may not be equivalent to factoring')."
- 1999-04-16 Patrick Juola: ". . . you're committing a grievious error if you think that something doesn't exist merely because you can't quantify it."
- 1999-04-16 Terry Ritter: "The issue is not the 'formalization' of something we know but cannot quantify, but rather something we actually do not know." ". . . you would have us assume that everyone knows that ciphers may be weak, and simply chooses to do nothing about it."
- 1999-04-17 ybizmt: "Nice rant. Where are you going with this and how does it sell your product?"
- 1999-04-17 Terry Ritter: "This is my bit for public education." "I have no modern products. I do offer cryptographic consulting time, and then I call it as I see it. I also own patented cryptographic technology which could be useful in a wide range of ciphers."
- 1999-04-18 John Savard: ". . . I think you can see why this design process actually *increases* the probability of a design which is strong against known attacks, but weak against a future attack someone might discover."
- 1999-04-18 John Savard: ". . . I do think that in the specific case of the AES, going back to the drawing board a bit would make quite a bit of sense. . . ."
- 1999-04-20 Terry Ritter: "You lost me on that one."
- 1999-04-21 John Savard: ". . . the cryptanalyst will find what he knows how to look for; and so, weaknesses beyond the reach of current cryptanalysis won't be found . . . ."
- 1999-04-21 Terry Ritter: "I claim the main use of cryptanalysis is in the give and take of a design process, not the end game of certification, which is what cryptanalysis cannot do."
- 1999-04-17 Patrick Juola: "I don't have *proof* that Pittsburgh won't be hit by a hurricane, but I can produce lots and lots of justification."
- 1999-04-17 Terry Ritter: "Which means to me that you have some understanding of the risk of hurricanes in Pittsburgh. You get this understanding from reported reality." "Unfortunately, neither you nor anyone else can have a similar understanding of the risk of cipher failure -- there is no reporting of cipher failure. There is instead every effort made to keep that information secret, and in fact to generate false reporting to buoy your unfounded delusion of strength."
- 1999-04-01 John Savard: ". . . what do we do if there is no way to obtain a lower bound on the strength of a cipher?" "Then it does make sense to look at the upper bound . . . ."
- 1999-04-20 Terry Ritter: "No. Completely false. I see no reason why the upper bound should have any correlation at all to the lower bound."
- 1999-04-21 John Savard: "Any security audit will have to include a disclaimer that the true security of the cipher systems used is essentially unknowable, but even real-world financial audits do routinely include various sorts of disclaimer."
- 1999-04-21 Terry Ritter: "We don't have financial disclaimers which say that the audit is 90 percent certain to be correct, which is the sort of thing you might like to think that cryptanalytic certification could at least do . . . ."
- 1999-04-21 John Savard: "Precisely because . . . we don't have a way to put a good lower bound on the effort required to break a cipher, I find it hard to think that I could achieve the goal . . . of making it 'as simple as possible, but no simpler'"
- 1999-04-17 wtshaw: "It's aways difficult to stop a wave, be it composed of hoards of combatants or algorithms."
- 1999-04-19 John Savard: "I agree with you that we don't have a way to prove that a cipher really is strong. But cryptanalysis still gives the best confidence currently available."
- 1999-04-20 Terry Ritter: "I guess I dispute 'confidence.' Confidence and Trust and Reliability are exactly what we do not have. I cannot say it more clearly: cryptanalysis gives us no lower bound to strength." "'Never underestimate your opponent.'"
- 1999-04-20 Mok-Kong Shen: "'There are no provable lower bounds for the amount of work of a cryptanalyst analyzing a public-key cryptosystem.'"
- 1999-04-23 Bryan G. Olson; CMSC (G): "This impression of the academic crypto community as a closed club that ignores the work of outsiders is flat out false." "The various attacks I've heard on academics are invariably by those whose work is simply not of the same caliber."
- 1999-04-23 John Savard: ". . . cryptography, like many other fields, has its cranks and quacks. "However, I don't think it's appropriate to automatically conclude that everyone who expresses concern about the way in which the public cryptography field is going is necessarily a crank."
- 1999-04-25 Bryan G. Olson; CMSC (G): "My counter is not the crypto community is right to generally ignore outsiders, but that in fact they do no such thing." "There is a situation worse than having all one's eggs in one basket." "What's worse is a system in which any one of many possible failures would be catastrophic."
- 1999-04-25 Sundial Services: "If you look at the design of these Feistel ciphers, well, to me they smack of Enigma, with its clockwork-like rotation of the cipher elements which ultimately proved its downfall."
- 1999-04-25 Sundial Services: "The layers in SIGABA are not all the same design. The layers in an n-round Feistel cipher are, literally by definition, all the same."
- 1999-04-25 Matthias Bruestle: "a 200kbit key would require about . . . 10 CPU hours. Would you want to wait 10 hours to read an email?"

- 1999-04-25 wtshaw: ". . . using several algorithms, it is reasonable to maximize security available not by living in fear of the weakest algorithm but working to make sure the strongest was included."
- 1999-04-24 wtshaw: ". . . specific information between individuals is not apt to pass but once or few times at the most."
- 1999-04-16 Medical Electronics Lab: "Some things which look really simple can have very complex mathematical relationships, and that's far more useful to a cipher design than something which appears complex but has a simple mathematical construction."
- 1999-04-17 wtshaw: "Then there are those designs that tend to impress people because they are overly complex in construction and perhaps made so for devious purposes."
- 1999-04-16 Boris Kazak: ". . . most published attacks against various ciphers are essentially attacking not as much the cipher per se, as its key schedule." "As a layman, I experimented with modular multiplication mod 2^32-1 and mod 2^32+1 . . . ." "BTW, I also experimented with multiplication mod 2^64+1 and 2^64-1."
- 1999-04-19 Jerry Coffin: ". . . 2^32-1 happens to be a prime number." "By contrast, 2^64-1 is what you might call extremely composite . . . ."
- 1999-04-19 Boris Kazak: "Sorry, 2^32-1 = 3*5*17*257*65537, but I have found nice ways to set up key-derived multipliers in this field. The maximum length of the multiplicative cycle is 65536 . . . ."
- 1999-04-20 Jerry Coffin: "Oops -- my bad. It's 2^31-1 which is a prime."
- 1999-04-20 Boris Kazak: "It is one of my 'essays' on the subject of *drunken* ciphers, where you set up a lot of S-boxes deriving them from the key, and then encrypt using the plaintext-dependent path through these S-boxes."
- 1999-04-20 Paul Rubin: "2^32-1 = . . . = 65537 *257 *17 *5 *3"
- 1999-04-17 wtshaw: ". . . cooperating to plow and plant the same furrow does not make lots of sense."
- 1999-04-17 Bruce Schneier: "Operations from the RISC subset are efficient on a wide variety of microprocessors."
- 1999-04-18: John Savard "Think FROG."
- 1999-04-18 wtshaw: "As I remember, the final pick is to be submitted to higher, political, authority for *approval*, which is apt not to be a technical decision based on purely scientific considerations."
- 1999-04-18 wtshaw: "You know, something easy to do knowing the key, and impractical not knowing it."
- 1999-04-18 wtshaw: "End decisions are to be made by political cronies."
- 1999-04-17 David A Molnar: "Most of the provable security I've seen comes in terms of statements like 'if this scheme is breakable, then factoring integers is easy . . . .'"
- 1999-04-17 Douglas A. Gwyn: "Simpler systems are, usually, easier to analyze more thoroughly. The more thoroughly we understand a class of systems, the more confident we can be that other analysts won't find some shortcut."
- 1999-04-17 Douglas A. Gwyn: "That is the reason for 'tiger teams', who act the part of bad guys. If your system hasn't been attacked by cryptanalysts who know *how* to mount such an attack, then it hasn't undergone sufficient Quality Control."
- 1999-04-17 Terry Ritter: "Even so, we still don't know that their guys aren't better, or even just luckier." ". . . if the goal was to get the best cipher, we would see a post-analysis re-design phase intended to fix known problems, with the final comparison being made between full-strength designs."
- 1999-04-18 wtshaw: "It would seem that the best use of the talent involved in the AES process would be to identify any new techniques that might have crept in, and see how these could be used to supplement the best of whatelse we know."
- 1999-04-17 Douglas A. Gwyn: "How could the design be conveyed to the implementor, then?"
- 1999-04-18 Douglas A. Gwyn: "The technical decision would already have been made, and any further process would be simply an approve/disapprove decision."
- 1999-04-18 H. Ellenberger: "Should we therfore stop analyzing existing ciphers?"
- 1999-04-18 Terry Ritter: "No. We should stop depending on any small set of ciphers, no matter how well analyzed."
- 1999-04-17 dscott@networkusa.net: ". . . you are right that most encryption methods use operations that are readily reversible. That is one reason I use as large an S-table that is possible in my encryption programs."
- 1999-04-18 wtshaw: ". . . the one thing that you highlight is something that perhaps is the biggest weakness in the utility of what you have done."
- 1999-04-18 dscott@networkusa.net: ". . . wrapped PCBC would add the 'all or nothing logic' to the AES candidates as well as solve the problem of handling files or sub files that are not made up of a number of bits that are a muliply of the block cipher size used."
- 1999-04-18 wtshaw: "There may be a fix to what you have done, making your encryption more acceptable to me, but you may not like the idea at first."
- 1999-04-19 dianelos@tecapro.com: "Whether you use simple or complex operations to describe a cipher is not relevant: a multiplication can be seen as a sequence of SHIFTs and ADDs . . . ."
- 1999-04-19 John Savard: "I think he was asking why block ciphers weren't more like FROG!"
- 1999-04-21 SCOTT19U.ZIP_GUY: "I think if one used a fixed block size of any size say 512 bytes. Then it would be easyer to study the cipher for its weak points."
- 1999-04-22 tomstdenis@my-dejanews.com: "If you had for example 24 rounds you could have 6 different P and S functions, each with 4 stagered rounds each."
- 1999-04-22 dianelos@tecapro.com: "I have often expressed the opinion that the biggest security risk in cipher design is the possible discovery of a catastrophic attack method against cipher designs considered strong today."
- 1999-04-22 John Savard: ". . . if we consider acceptable a design that was initially weak against known attacks, but which was modified later specifically to resist them, then we are . . . more likely to fall prey to *new* attacks than we would be with a design that was resistant to the known attacks right from the start."
- 1999-04-22 SCOTT19U.ZIP_GUY: "The AES contest is not about having secure encryption. The NSA would never allow a good common method to be blessed by the government for general use."
- 1999-04-24 wtshaw: "The best thing to be is open and honest, something government does not particularly like to do as it limits arbitrary power."
- 1999-04-25 wtshaw: ". . . if you only learn a few selected crypto primatives, those are the ones you are apt to utilize."
- 1999-04-25 bryan.olson@uptronics.com: "I (Bryan) only wrote that in quoting its author, John Savard."
- 1999-04-26 dianelos@tecapro.com: ". . . a good variable cipher protocol would hide the information about which cipher or cipher combination has been used."
- 1999-04-26 Jerry Coffin: ". . . when you combine two algorithms, you're basically designing a new cypher. If you're lucky, it'll combine the strengths of both the base cyphers, while negating some of the weaknesses of each."
- 1999-04-27 Trevor Jackson, III: "Can you show a real example? Are there *any* known weaknesses in combining any pair of the following: Blowfish, IDEA, 3DES?"
- 1999-04-27 Jerry Coffin: "Of course there are real facts. I've demonstrated how one trivial example shows up poorly . . . ."
- 1999-04-28 dianelos@tecapro.com: "Cascading ciphers does increase security in this sense. Schneier discusses this method in chapter 15.7 of the second edition of Applied Cryptography. There are also two papers by Maurer discussing the combination of block ciphers and of stream ciphers."
- 1999-04-17 Douglas A. Gwyn: ". . . if your tiger team is really good and experienced . . ., you can attain a certain degree of confidence based on their positive evaluation . . . ."
- 1999-04-18 Uri Blumenthal: "At least one publicly known cipher (GOST) uses ADD (a more complex operation)..."
- 1999-04-17 Douglas A. Gwyn: ". . . treating the whole message would be a problem in many applications where data is generated in a stream, for example a TELNET session."
- 1999-04-17 Geoff Thorpe: ". . . good ideas can quickly get buried in slop, or just unmasked as reactionary drivel as they seem (IMHO) to in Mr Ritter's post below ..." "I disagree - your point of view has some merit but is no more valid than the polar opposite statement." ". . . thousands of hobbyists and professionals all doing their damndest to break each others ciphers gives me personally some confidence in the value of 'standing the test of time'."
- 1999-04-17 Terry Ritter: "The strength value measured on academics cannot apply to the real problem." "I have shown several different approaches which are helpful for security even in an environment where we cannot assure ourselves of the strength of any particular cipher." "You do not know the probabilities, and you do not know the risk, yet you would have us manage the situation using exactly these quantities. That is mad." ". . . our Opponents operate in secrecy. That means we actually do not know when our ciphers fail. But unless we know about failure, we cannot assess risk." ". . . it is openly argued that 'uncertified' ciphers have more risk, without being able to quantify that risk. While I would hope every cipher would get as much analysis as it could get, the 'certification' of one cipher does not give us what we need."
- 1999-04-18 Douglas A. Gwyn: "Just so you know, I appreciate your work and especially your making useful information available via the Web."
- 1999-04-18 Terry Ritter: "Coming from you, that means a lot. Thanks."
- 1999-04-20 John Savard: "I will take the opportunity to acknowledge both that you have made contributions through your own work, as well as by representing a point of view that points in the direction of what I, also, feel is a correction needed by the cryptographic community."

- 1999-04-20 Terry Ritter: "I think it should be very disturbing to anyone actually trying to do Science to have to consider whether or not the conclusions they come to are a 'minority position.'"
- 1999-04-18 ovbxotm: "Guess its time to add an entry to my killfile."
- 1999-04-18 Geoff Thorpe: "Abstract, extremist, conspiracy-theoretic poppycock." "I don't agree but I can't PROVE you wrong. It does not mean you're right - and many people share MY point of view on this." "Do you insure your car? Do you know the probabilities or the risk? I take a look at my driving, most others' driving, the risk . . . and make a judgement call." "I certainly have no time for such cold-war style conspiracy theories - that somehow the knowledge present in military/intelligence agencies . . . is probably so . . . disimilar to what is available in the open as to make work, developments, and failures (to break ciphers) in the open completely irrelevant . . . ."
- 1999-04-19 Douglas A. Gwyn: "For many perfectly decent people, their own governments can become their enemies; history gives us many instances of this. "While the cryptanalytic bureaus of most third-world countries might not be very advanced, the one in the US certainly is."
- 1999-04-19 Geoff Thorpe: ". . . I was just taking issue with what I perceived to be the following idea: Until we actually break it, or *prove* it secure, we have no more measure of strength for it than for another (less 'investigated') one. I feel that quite the opposite is true - it IS a very appropriate statistical measure of strength, and moreover the only realistic one we have to work with."
- 1999-04-19 Terry Ritter: "In summary: 1) We cannot estimate the probability that an effective attack exists which we did not find; and 2) We cannot estimate the probability that even if such an attack does exist, our Opponents can find it and use it. I thus claim that we CAN know nothing of the probability of future cipher failure, and cannot even reason that this probability is 'small.' The practical consequence of this is that we cannot trust any cipher." "With respect to the problem of potential catastrophic failure from a single-cipher system, no amount of cryptanalysis can prevent such failure. Both untested ciphers and massively-tested ciphers are the same in the sense that neither can be trusted."
- 1999-04-21 Earth Wolf: "I'll trust DES a heck of a lot more than I trust ROT-13. And I'll trust 3DES a heck of a lot more than I trust DES." ". . . basic cryptography involves making your best estimate of your opponents' capabilities and desiging a cipher which, to the best of your knowledge, will be impervious to those capabilities for as long as it needs to be."
- 1999-04-21 Terry Ritter: "My position is that no rational extrapolation of past tests to future strength is possible. The lack of literature containing such a thesis is consistant with my position, and inconsistant with the alternative." "My guess would be that you "trust" DES because nobody has openly demonstrated that they can break it. So if you worry that your information will be stolen by academics, you *can* have some reasonable degree of trust in DES. "But if you use cryptography to protect your information from those who operate in secret and hide their successes, you have no data upon which to base trust." "Basic *social* cryptography . . . must concern itself with every non-military use for hiding data. "From this I conclude that the use of a single standard cipher throughout society would be an unthinkable risk."
- 1999-04-21 John Savard: "Here, you and I are in agreement. New attacks are being found against symmetric block ciphers, such as the boomerang attack and the slide attack. Also, one of the papers on the NIST site is called 'Future Resiliency', and it is a defense of that point of view."
- 1999-04-23 Douglas A. Gwyn: "The Japanese cleverly misled our intelligence analysts into believing that their fleet was still in home waters."
- 1999-04-19 Douglas A. Gwyn: "Generally speaking, Terry is right to be concerned over the unknown, but some risks are greater than others. The specific algorithms you mentioned previously are among the better risks."
- 1999-04-19 David A Molnar: ". . . it was known that knapsack public-key systems leaked bits of information long before any specific catastophic results were known. The single bit doesn't help much, but acts as a warning sign that something is wrong."
- 1999-04-20 Trevor Jackson, III: "Layered algorithms do not dictate expensive or complex operational requirements. The implementation of a layered cipher needs some care, but no more than any other secure system. This issue appears to be a red herring."
- 1999-04-20 Geoff Thorpe: "I may not design ciphers but I can definately slug it out with most people regarding probability theory, statistics, and logic. I also have to assist with various API designs and have been on the (l)using end of quite a few if we want to talk standards, picking algorithms, and covering butts (oh yeah, I've done quite a bit of Risk Management related stuff too)."
- 1999-04-20 Terry Ritter: "Recall that my position does not rest upon an estimation of someone else's capabilities. It is not my *opinion* that any cipher we have *might* possibly break -- that is fact. I assume the worst case, and propose systems to provide strength even then. "Your position, dare I state it, is that you *can* estimate the capabilities of your Opponents. You also say you can estimate the future strength of a cipher from past tests. But for all this claiming, we see no similar statements in the scientific literature. So these are simply your opinions, and I see no supporting facts."
- 1999-04-20 Geoff Thorpe: ". . . you assume the worst case. Whilst you certainly will never be accused of lacking precaution, why should I accept that your position the only appropriate one to adopt?" ". . . I still place a lot of stock in what *I* rank as ciphers of tested strength and wouldn't want any system of mine having too many 'new toy' ciphers creeping in." ". . . I have already said that my view (that a cipher not falling over despite some considerable efforts against it does merit some 'value') is not based on any exact science." "And in case you ask, no - I know of NO research paper to support this and have no interest in attempting to create some when I'm already satisfied."
- 1999-04-21 Terry Ritter: "In this case I agree with virtually the entire body of cryptanalytic literature in that one CANNOT know cipher strength. I also think it is fruitless to speculate on strength, or on the capabilities of our Opponents, and that we are better of spending our time protecting against failures which cryptanalysis cannot avoid." ". . . my 'contrary opinion' -- that the past history of the strength of a cipher does NOT tell us about its future strength -- again reflects the scientific literature. I am aware of no articles at all that show such a correlation. That is not *my* opinion, that is the prevailing scientific understanding." "In crypto, we do not have the same cues which support our understanding of risk in real life." "My point is precisely that cryptanalysis ("breaking") *cannot* tell us if a cipher is weak. My point is that we must assume weakness *without* having to break the cipher, if failure would be disastrous. Since that is my point, I hardly need do the opposite to make my argument." "I think unscientific arguments *would* be called 'incorrect.' You assume something trivial like extrapolating the strength of a cipher from its cryptanalytic testing -- something which does not exist in the scientific literature. "In contrast, I assume that any cipher may fail -- and this is the nearly universal scientific understanding. I would call that correct." "It is more than a matter of disagreement; it is a matter of you being wrong." "Will you find a cipher designer who will say in so many words that he or she *can* predict the strength of a cipher based on its cryptanalytic history? Will you find someone who will say they can predict the capabilities of future Opponents based on current academic results? I don't think so." "I support all the cryptanalysis we can get. But, in the end, we can trust neither cipher to remain unbroken. So, in that sense, new and old ciphers are both equal in their untrustability."
- 1999-04-21 Geoff Thorpe: ". . . if you are right, you need to find a better way of understanding my view and showing me constructively why it is definitively wrong if you want to get anywhere."
- 1999-04-22 John Savard: "While there will be pressure to adopt the standard ciphers for interoperability, such a system definitely does allow for a new cipher to become a new part of the set."
- 1999-04-20 Jim Gillogly: "I think Terry Ritter's right to be concerned about having essentially everyone move to a single new cipher." "I also agree with him that a careful concatenation of ciphers can help limit the damage." "However, I (like John Savard) think Terry overstates some issues." "These and similar remarks suggest that a conservative threat analysis must regard the opponents as god-like in their cryptanalytic capabilities."
- 1999-04-21 Terry Ritter: "As far as I can recall, the only people I have accused of being 'god-like' are the crypto gods who seem to be able to predict: 1) the future strength of a cipher, based on past tests; and 2) the capabilities of unknown Opponents, based on the capabilities of known academics." "But what *are* we to assume? Even a *modest* "value" for Opponent capabilities is also "not useful" to us." "So we first don't know the difficulty of the problem, and then don't know the capabilities our Opponents can bring to the solution." "I suggest that this whole line of inquiry (into cipher strength and Opponent strength) is a waste of time."
- 1999-04-20 Jim Gillogly: ". . . you can't expect to get expert analysis for free... ."
- 1999-04-21 Terry Ritter: ". . . you should agree that each user should have a similar power to make their own choices of cipher. That sounds just fine to me."
- 1999-04-21 Jim Gillogly: "We have wider areas of agreement than disagreement; I'm happy to leave it at that."
- 1999-04-21 Trevor Jackson, III: "Some clarification may be called for in that your statements can be construed as claims that cipher diversity solves the problem of inferior talent/resources/etcetera with respect to dark-side adversaries and future adversaries of all shades. I believe this absolutist position to be false."
- 1999-04-21 Terry Ritter: "I am not sure that I have made such a claim, which I also think is false." "I don't know what could be clearer than my repeated statement that we can trust no cipher. Cipher use cannot build trust in strength. Cryptanalysis cannot certify strength. We must always be aware that failure is a possibility, and we are even unable to estimate that probability. When the consequences of cipher failure are catastrophic, we simply cannot afford to depend on any one cipher."
- 1999-04-21 Trevor Jackson, III: "There's no need to assume perfection or god-like omniscience to motivate as conservative an approach as possible."
- 1999-04-20 Trevor Jackson, III: "Given that there are few ciphers that have survived "the gauntlet" for a respectable period of time compared to the many ciphers without that maturity, the odds look to me much as Ritter described them. If I pick a young cipher, it may be broken tomorrow. If I pick an elderly cipher it may be broken tomorrow." "The situation we face is that we have dozens of reasonably strong ciphers, whose relative strengths are immeasurable. We may theorize about their relative merits, but we can't measure their strength in any fundamental sense. Given this I believe it makes sense to reduce risk of catastrophic failure by using a composite of the strongest ciphers of which we are aware."
- 1999-04-25 H. Ellenberger: "If science could *prove* that a certain feasible algorithm is unbreakable, everybody could use it and there would be no need for annother one (and probably this algorithm would be a intelligent combination of various principles)." "Without such a proved algorithm it is indeed too risky to use a single algorithm. It is exposed to focused attacks from all sides, and in case it sould break, damages are too important."
- 1999-04-25 Geoff Thorpe: "I don't know if you were following the thread but I was just saying that certain established and unbroken (despite plentiful attempts) ciphers are worthy of more 'trust', 'strength', 'tested strength', 'confidence' or whatever than new-toy ciphers."
- 1999-04-19 Terry Ritter: "I spent some hours responding to this huge article, and only at the end realized (I assume correctly) that most of the controversy was about something which I consider peripheral to the main issue." "My main argument starts out that no matter how much analysis is done, there is always the *possibility* that a cipher may fail anyway. I doubt anyone disagrees with this." "Since cipher failure is *possible*, we need to look at the consequences of failure: If this is to be the one standard cipher for society, the results of such failure would be *catastrophic*. Again, hardly controversial stuff." "We can do something about this: We can innovate various procedures and protocols to avoid single-cipher failure. As a general concept, it is hard to imagine that even *this* is controversial. The actual technical details, of course, are arguable and changeable." "The package I have proposed includes compartmentalizing our data under different ciphers, thus reducing the amount of data at risk of any single cipher failure. (This virtually requires us to change ciphers frequently.) I also proposed multi-ciphering as a matter of course (reducing the likelihood of failure), and having a growing body of ciphers from which to choose. Other proposals can of course be made."

- 1999-04-19 Trevor Jackson, III: "I rise (type) to speak in support of the proposition." "The tendency of the market to focus on a single (or few) best product(s) is well established. The true operational basis for this is most often simple laziness."
- 1999-04-19 Gurripato (x=nospam): "There must be some reason why a product is well established, not necessarily quality."
- 1999-04-20 Trevor Jackson, III: "You are certainly free to disagree with the marketing theory that says mature markets are better. After all, I disagree with it too." "I find it especially unsuitable for the field of crypto."

Subject: Re: Announce - ScramDisk v2.02h
Date: Fri, 02 Apr 1999 19:03:38 GMT
From: ritter@io.com (Terry Ritter)
Message-ID: <3705141b.4312658@news.io.com>
References: <7e2vge$o82$1@nnrp1.dejanews.com>
<7e2br4$7i3$1@nnrp1.dejanews.com>
Newsgroups: alt.security.pgp,sci.crypt
Lines: 58

On Fri, 02 Apr 1999 17:41:35 GMT, in
<7e2vge$o82$1@nnrp1.dejanews.com>, in sci.crypt ssimpson@hertreg.ac.uk
wrote:

>[...]
>The point of that section of the document was that an adversary is not aware
>of which algorithm you use....They have no method of detecting whether TEA,
>Blowfish, IDEA, 3DES etc is used. Both PGPDisk & Bestcrypt plainly state the
>algorithm employed.
>
>So, to "brute force" a ScramDisk container an adversary has to effectively
>try all 10 ciphers, whereas to brute force other products containers they
>only have to try 1 cipher. Is this snake oil? No.

For some years I have been promoting the idea of using multiple
ciphers, but my argument is different:

1. I see little keyspace (brute-force search) advantage with just a
few ciphers. If we had a robust industry of replaceable cipher
modules, with tens of thousands of possibilities and growing all the
time, *then* we get some keyspace. But with just 10 ciphers, the
keyspace advantage is lost in the noise of attacks which need 2**43
known-plaintexts.

2. The big advantage of having a huge number of ciphers is the burden
it places on any Opponent, who necessarily must keep up. Opponents
must distinguish each cipher, obtain it, break it, then construct
software and perhaps even hardware to automate the process. Given a
continuous production of large numbers of new ciphers, I believe that
"keeping up" must have a terrible cost that not even a country can
afford.

3. The risk of using a single popular cipher (no matter how
extensively analyzed) is that a vast amount of information is
protected by one cipher. This makes that cipher a special target -- a
contest with a payoff far beyond the games we normally play. I think
we want to avoid using such a cipher.

4. To make the cost of multiple ciphers real, we cannot keep using
the same cipher, but instead must use different (new) ciphers
periodically. We will want to use the same cipher-system, so our
system must support "clip-in" modules for ciphers which have not yet
been written.

5. One of the facts of ciphering life is that we cannot prove the
strength of any cipher. Even NIST review and group-cryptanalysis does
not give us proven strength in a cipher, so any cipher we select might
be already broken, and we would not know. We cannot change this, but
we can greatly improve our odds as a user, by multi-ciphering under
different ciphers. Doing this means an Opponent must break *all* of
those ciphers -- not just one -- to expose our data. I like the idea
of having three layers of different cipher, each with its own key.

Terry Ritter ritter@io.com http://www.io.com/~ritter/
Crypto Glossary http://www.io.com/~ritter/GLOSSARY.HTM

Subject: Re: Announce - ScramDisk v2.02h
Date: Fri, 02 Apr 1999 18:37:41 -0500
From: Boris Kazak <bkazak@worldnet.att.net>
Message-ID: <370554C5.156E@worldnet.att.net>
References: <3705141b.4312658@news.io.com>
Newsgroups: alt.security.pgp,sci.crypt
Lines: 65

Terry Ritter wrote:
>

> >The point of that section of the document was that an adversary is not aware
> >of which algorithm you use....They have no method of detecting whether TEA,
> >Blowfish, IDEA, 3DES etc is used. Both PGPDisk & Bestcrypt plainly state the
> >algorithm employed.
> >
> >So, to "brute force" a ScramDisk container an adversary has to effectively
> >try all 10 ciphers, whereas to brute force other products containers they

> >only have to try 1 cipher. Is this snake oil? No.
> /...../
> 2. The big advantage of having a huge number of ciphers is the burden
> it places on any Opponent, who necessarily must keep up. Opponents
> must distinguish each cipher, obtain it, break it, then construct
> software and perhaps even hardware to automate the process. Given a
> continuous production of large numbers of new ciphers, I believe that
> "keeping up" must have a terrible cost that not even a country can
> afford.

And how about a "variable" cipher? The one which philosophy will be based on

1. a *big* number of key-derived S-boxes
2. a plaintext-dependent sequence of invocation for these S-boxes.

In case where there will be enough plaintext-dependent variability, no two plaintexts will be encrypted along the same sequence. This will be essentially equivalent to adding the plaintext space and the key space together. With all the additional effort for the attacker.

>
> 3. The risk of using a single popular cipher (no matter how
> extensively analyzed) is that a vast amount of information is
> protected by one cipher. This makes that cipher a special target -- a
> contest with a payoff far beyond the games we normally play. I think
> we want to avoid using such a cipher.
>
> 4. To make the cost of multiple ciphers real, we cannot keep using
> the same cipher, but instead must use different (new) ciphers
> periodically. We will want to use the same cipher-system, so our
> system must support "clip-in" modules for ciphers which have not yet
> been written.

A long overdue problem - standard cipher-to-application interface.

> 5. One of the facts of ciphering life is that we cannot prove the
> strength of any cipher. Even NIST review and group-cryptanalysis does
> not give us proven strength in a cipher, so any cipher we select might
> be already broken, and we would not know. We cannot change this, but
> we can greatly improve our odds as a user, by multi-ciphering under
> different ciphers. Doing this means an Opponent must break *all* of
> those ciphers -- not just one -- to expose our data. I like the idea
> of having three layers of different cipher, each with its own key.
>

Have a little pity for the export bureaucrats! They are kind enough to allow exporting three different ciphers, and now you want to use their kindness to show all the world that their export regulations can be circumvented and that they are not worth the paper they are printed on.

> ---
> Terry Ritter ritter@io.com http://www.io.com/~ritter/
> Crypto Glossary http://www.io.com/~ritter/GLOSSARY.HTM

Best wishes BNK

Subject: Re: Announce - ScramDisk v2.02h
Date: Sun, 04 Apr 1999 03:46:48 GMT
From: lyeoh@pop.jaring.nospam.my (Lincoln Yeoh)
Message-ID: <3706da06.2109438@nntp.jaring.my>
References: <3705141b.4312658@news.io.com>
Newsgroups: alt.security.pgp,sci.crypt
Lines: 47

On Fri, 02 Apr 1999 19:03:38 GMT, ritter@io.com (Terry Ritter) wrote:

>5. One of the facts of ciphering life is that we cannot prove the
>strength of any cipher. Even NIST review and group-cryptanalysis does
>not give us proven strength in a cipher, so any cipher we select might
>be already broken, and we would not know. We cannot change this, but
>we can greatly improve our odds as a user, by multi-ciphering under
>different ciphers. Doing this means an Opponent must break *all* of
>those ciphers -- not just one -- to expose our data. I like the idea
>of having three layers of different cipher, each with its own key.

Yeah. I like the idea of superencryption too, and I don't know why so few people seem to like it. So far I have not had a good answer to how an attacker would know if he or she has succeeded.

In fact I'd go further- by having layers of many and not just 3 layers of encryption, and each layer not leaking any headers at all, the attacker's work will be very much harder, even if each layer is just 40 bit crypto.

I would think if we use ciphers A, B, C in such a headerless manner, A.B.C is unlikely to be weaker than either A or B or C alone, despite some FUD about "cipher interaction", heck if such an effect was likely we'd see more cryptographers encrypting stuff with various other ciphers to decrypt it.

Since no one can prove a single cipher is secure, it is hubris to assume that one cipher is better than the rest or that one can even select a single preferred one. For high confidentiality one could perhaps super encrypt data with the top 6 AES contenders.

As for cpu consumption, I'm wondering if chaining reduced round ciphers together would be too risky. But your point about having a large pool of ciphers can help here. With the number of possibilities, analysis could be come prohibitive. The opponent may just have to resort to brute force - trying out various keys with various ciphers.

I'm wondering if cryptanalysis of plaintext encrypted with ABCDEFG be different from analysis of AGDCBEF?

Have fun!

Link.

****
Reply to: @Spam to
lyeoh at @people@uu.net
pop.jaring.my @
****

Subject: Re: Announce - ScramDisk v2.02h
Date: 9 Apr 1999 11:10:56 GMT
From: aph@cygnus.remove.co.uk (Andrew Haley)
Message-ID: <7ekn80$kc1$1@korai.cygnus.co.uk>
References: <3706da06.2109438@nntp.jaring.my>
Newsgroups: sci.crypt
Lines: 16

[Newsgroups list trimmed]

Lincoln Yeoh (lyeoh@pop.jaring.nospam.my) wrote:

: I like the idea of superencryption too, and I don't know why so few
: people seem to like it. So far I have not had a good answer to how
: an attacker would know if he or she has succeeded.

The answer is simple. Kerckhoff's maxim says that your attacker knows the cryptosystem you're using, but does not know the key. If you're using superencryption, your attacker knows which systems you're using.

Of course, your attacker must now analyze the compound cipher, which is almost certainly harder to do than than attacking a single cipher.

Andrew.

Subject: Re: Announce - ScramDisk v2.02h
Date: Fri, 09 Apr 1999 17:10:12 GMT
From: ritter@io.com (Terry Ritter)
Message-ID: <370e33ea.2675997@news.io.com>
References: <7ekn80$kc1$1@korai.cygnus.co.uk>
Newsgroups: sci.crypt
Lines: 39

On 9 Apr 1999 11:10:56 GMT, in <7ekn80$kc1$1@korai.cygnus.co.uk>, in sci.crypt aph@cygnus.remove.co.uk (Andrew Haley) wrote:

>[Newsgroups list trimmed]
>
>Lincoln Yeoh (lyeoh@pop.jaring.nospam.my) wrote:
>
>: I like the idea of superencryption too, and I don't know why so few
>: people seem to like it. So far I have not had a good answer to how
>: an attacker would know if he or she has succeeded.
>
>The answer is simple. Kerckhoff's maxim says that your attacker knows
>the cryptosystem you're using, but does not know the key. If you're
>using superencryption, your attacker knows which systems you're using.

That's fine if you always use the same ciphers in the same order. But if the ciphers are dynamically selected by keying, or just dynamically selected frequently by communications under cipher, the attacker does *not* know "which systems you're using." Kerckhoff's maxim does not apply.

I suggest that each communication include a small encrypted control channel, over which a continuous conversation of what ciphers to use next takes place. This would be an automatic negotiation, somewhat like occurs in modern modems, from cipher selections approved by the users (or their security people).

>Of course, your attacker must now analyze the compound cipher, which
>is almost certainly harder to do than than attacking a single cipher.

Yes. Even if each cipher used has known weaknesses, those may not be exploitable in the multi-ciphering case.

Terry Ritter ritter@io.com http://www.io.com/~ritter/
Crypto Glossary http://www.io.com/~ritter/GLOSSARY.HTM

Subject: Re: Announce - ScramDisk v2.02h
Date: 9 Apr 1999 13:16:35 -0500
From: juola@mathcs.duq.edu (Patrick Juola)
Message-ID: <7elg63eeq1@quine.mathcs.duq.edu>
References: <370e33ea.2675997@news.io.com>
Newsgroups: sci.crypt
Lines: 32

In article <370e33ea.2675997@news.io.com>, Terry Ritter <ritter@io.com> wrote:
>
>On 9 Apr 1999 11:10:56 GMT, in <7ekn80$kc1$1@korai.cygnus.co.uk>, in
>sci.crypt aph@cygnus.remove.co.uk (Andrew Haley) wrote:
>
>>[Newsgroups list trimmed]
>>
>>Lincoln Yeoh (lyeoh@pop.jaring.nospam.my) wrote:
>>
>>: I like the idea of superencryption too, and I don't know why so few
>>: people seem to like it. So far I have not had a good answer to how
>>: an attacker would know if he or she has succeeded.
>>
>>The answer is simple. Kerckhoff's maxim says that your attacker knows
>>the cryptosystem you're using, but does not know the key. If you're
>>using superencryption, your attacker knows which systems you're using.
>
>That's fine if you always use the same ciphers in the same order. But
>if the ciphers are dynamically selected by keying, or just dynamically
>selected frequently by communications under cipher, the attacker does
>*not* know "which systems you're using." Kerckhoff's maxim does not
>apply.

Unfortunately, this isn't the case.

If the system is dynamically selected by keying, then the exact
selection becomes part of the key. If you are taking a set of cyphers
and reordering them, Kerchoff's maxim suggests that you have to assume
that the attacker knows the set of cyphers and just doesn't know
the order.

-kitten

Subject: Re: Announce - ScramDisk v2.02h
Date: Fri, 09 Apr 1999 18:06:51 -0400
From: Boris Kazak <bkazak@worldnet.att.net>
Message-ID: <370E79FB.5B1C@worldnet.att.net>
References: <7elg63eeq1@quine.mathcs.duq.edu>
Newsgroups: sci.crypt
Lines: 27

Patrick Juola wrote:
>
> In article <370e33ea.2675997@news.io.com>, Terry Ritter <ritter@io.com>
wrote:
> >But
> >if the ciphers are dynamically selected by keying, or just dynamically
> >selected frequently by communications under cipher, the attacker does
> >*not* know "which systems you're using." Kerckhoff's maxim does not
> >apply.
>
> Unfortunately, this isn't the case.
>
> If the system is dynamically selected by keying, then the exact
> selection becomes part of the key. If you are taking a set of cyphers
> and reordering them, Kerchoff's maxim suggests that you have to assume
> that the attacker knows the set of cyphers and just doesn't know
> the order.
>
> -kitten

It all depends on the numbers in question.
  If there are 2,3,10, even 100 ciphers, you can afford an exhaustive
search, but what if there are 2^16 ciphers (or 2^16 variations of the
base cipher), what then?
  Essentially you are adding together the key space and the ciphers
space, with the corresponding increase of problems for an attacker.

Best wishes BNK

Subject: Re: Announce - ScramDisk v2.02h
Date: Fri, 09 Apr 1999 22:24:20 GMT
From: ritter@io.com (Terry Ritter)
Message-ID: <370e7e04.21648784@news.io.com>
References: <7elg63eeq1@quine.mathcs.duq.edu>
Newsgroups: sci.crypt
Lines: 47

On 9 Apr 1999 13:16:35 -0500, in <7elg63eeq1@quine.mathcs.duq.edu>,
in sci.crypt juola@mathcs.duq.edu (Patrick Juola) wrote:

>In article <370e33ea.2675997@news.io.com>, Terry Ritter <ritter@io.com> wrote:

>>
>>On 9 Apr 1999 11:10:56 GMT, in <7ekn80$kc1$1@korai.cygnus.co.uk>, in
>>sci.crypt aph@cygnus.remove.co.uk (Andrew Haley) wrote:
>>>[...]
>>>The answer is simple. Kerckhoff's maxim says that your attacker knows
>>>the cryptosystem you're using, but does not know the key. If you're
>>>using superencryption, your attacker knows which systems you're using.
>>
>>That's fine if you always use the same ciphers in the same order. But
>>if the ciphers are dynamically selected by keying, or just dynamically
>>selected frequently by communications under cipher, the attacker does
>>*not* know "which systems you're using." Kerckhoff's maxim does not
>>apply.
>
>Unfortunately, this isn't the case.
>
>If the system is dynamically selected by keying, then the exact
>selection becomes part of the key.

Which of course means that the dynamic selection is not subject to
Kerckhoff's maxim. End case 1.

>If you are taking a set of cyphers
>and reordering them, Kerchoff's maxim suggests that you have to assume
>that the attacker knows the set of cyphers and just doesn't know
>the order.

First of all, this is not true if we have a dynamically-expanding set
of ciphers. Every cipher is only "known" to the Opponents after they
have identified it, acquired it, analyzed it, and, presumably, broken
it. All this necessarily takes time, and this time works for the user
and against the attacker.

But even if The Opponents *do* know the set of existing ciphers, when
the ciphers used are selected by some periodic random selection, there
is secret information which is not exposed. This is a sort of keying.
We cannot simply assume that it *is* exposed. End case 2.

Terry Ritter ritter@io.com http://www.io.com/~ritter/
Crypto Glossary http://www.io.com/~ritter/GLOSSARY.HTM

Subject: Re: Announce - ScramDisk v2.02h
Date: Fri, 9 Apr 1999 11:31:40 -0700
From: "Harvey Rook" <redrook@someyahoo.com>
Message-ID: <7elhhc$ic6@news.dns.microsoft.com>
References: <370e33ea.2675997@news.io.com>
Newsgroups: sci.crypt
Lines: 65

Terry Ritter <ritter@io.com> wrote in message
news:370e33ea.2675997@news.io.com...
>
> On 9 Apr 1999 11:10:56 GMT, in <7ekn80$kc1$1@korai.cygnus.co.uk>, in
> sci.crypt aph@cygnus.remove.co.uk (Andrew Haley) wrote:
>
> >The answer is simple. Kerckhoff's maxim says that your attacker knows
> >the cryptosystem you're using, but does not know the key. If you're
> >using superencryption, your attacker knows which systems you're using.
>
> That's fine if you always use the same ciphers in the same order. But
> if the ciphers are dynamically selected by keying, or just dynamically
> selected frequently by communications under cipher, the attacker does
> *not* know "which systems you're using." Kerckhoff's maxim does not
> apply.
>
This is incorrect. By Kerchhoff's maxim, you have to assume your attacker
has a copy of your deciphering machine. If he has a copy of your deciphering
machine, the attacker can figure out the algorithm you use to select
ciphers.
Once he knows the algorithm used to select ciphers, super-encipherment
only doubles or triples the amount of time needed to brute force. You'd
be much better off adding an extra byte to your key.

> I suggest that each communication include a small encrypted control
> channel, over which a continuous conversation of what ciphers to use
> next takes place. This would be an automatic negotiation, somewhat
> like occurs in modern modems, from cipher selections approved by the
> users (or their security people).
>
> >Of course, your attacker must now analyze the compound cipher, which
> >is almost certainly harder to do than than attacking a single cipher.
>
> Yes. Even if each cipher used has known weaknesses, those may not be
> exploitable in the multi-ciphering case.
>
It's only harder by one or two orders of magnitude. Computers built 3 years
from now
will have enough power to compensate for this difference. Adding an extra
byte
to your key makes the problem harder by 4 to 8 orders of magnitude. This is
much

harder to attack, yet it's simpler and cleaner to analyze. Simple clean systems, are
much less likely to have holes.

Unexpected weaknesses in ciphers designed by good cryptographers (Say Rivest, or Schneier)
are very unlikely to appear. Remember DES, after 25 years of analysis, is still only
vulnerable to a brute force attack. I'd be willing to bet that RC-6 and TwoFish will withstand
the same scrutiny.

Security breaks down because of bad passwords and poor protocols. Not because of cipher
weakness. Plan your system accordingly, or you are deluding yourself.

Harv
RedRook At Zippy The Yahoo Dot Com
Remove the Zippy The to send email.

Subject: Re: Announce - ScramDisk v2.02h
Date: Fri, 09 Apr 1999 22:24:44 GMT
From: ritter@io.com (Terry Ritter)
Message-ID: <370e7e29.21686311@news.io.com>
References: <7elhhc$ic6@news.dns.microsoft.com>
Newsgroups: sci.crypt
Lines: 104

On Fri, 9 Apr 1999 11:31:40 -0700, in
<7elhhc$ic6@news.dns.microsoft.com>, in sci.crypt "Harvey Rook"
<redrook@someyahoo.com> wrote:

>Terry Ritter <ritter@io.com> wrote in message
>news:370e33ea.2675997@news.io.com...
>>
>> On 9 Apr 1999 11:10:56 GMT, in <7ekn80$kc1$1@korai.cygnus.co.uk>, in
>> sci.crypt aph@cygnus.remove.co.uk (Andrew Haley) wrote:
>>
>> >The answer is simple. Kerckhoff's maxim says that your attacker knows
>> >the cryptosystem you're using, but does not know the key. If you're
>> >using superencryption, your attacker knows which systems you're using.
>>
>> That's fine if you always use the same ciphers in the same order. But
>> if the ciphers are dynamically selected by keying, or just dynamically
>> selected frequently by communications under cipher, the attacker does
>> *not* know "which systems you're using." Kerckhoff's maxim does not
>> apply.
>>
>This is incorrect. By Kerchhoff's maxim, you have to assume your attacker
>has a copy of your deciphering machine. If he has a copy of your deciphering
>machine, the attacker can figure out the algorithm you use to select
>ciphers.

That is no more true than saying that the attacker can figure out the
message key or the initialization vector. We assume the system has
some reliable random source to make such values. Similar values
select what ciphers to use in what order. And this should change
frequently.

>Once he knows the algorithm used to select ciphers, super-encipherment
>only doubles or triples the amount of time needed to brute force. You'd
>be much better off adding an extra byte to your key.

The issue is far beyond increasing the keyspace, since that would take
a huge number of ciphers. The issue instead is about spreading
messages among many different ciphers, thus forcing The Opponents to
"keep up" by identifying, acquiring, attacking, and breaking a
continuing flow of new ciphers. This forces The Opponents to invest
far more, and breaking any of these gains them far less. Note the
contrast to the current situation, where only a few main ciphers are
used, so only those need be broken.

>>[...]
>> Yes. Even if each cipher used has known weaknesses, those may not be
>> exploitable in the multi-ciphering case.
>>
>It's only harder by one or two orders of magnitude.

Well, let's see: Suppose we have about 4K ciphers. By selecting 3 at
random so we have about 4k**3 selections. This is somewhat larger
than "two orders of magnitude."

But, again, the issue is not really the increase in keyspace. The
issue is that The Opponents have to increase their analysis budget by,
say, a factor of 1k. And even when they are successful, they get at
most 1 message out of 1k. And we reduce that to 1 in 1k*1k*1k by
using 3 layers of cipher.

>Computers built 3 years
>from now

>will have enough power to compensate for this difference. Adding an extra
>byte
>to your key makes the problem harder by 4 to 8 orders of magnitude. This is
>much
>harder to attack, yet it's simpler and cleaner to analyze. Simple clean
>systems, are
>much less likely to have holes.

Using a layered structure is far, far simpler than trying to analyze a cipher to the degree of predicting that one cipher is stronger than another by n orders of magnitude. Neither value is known, so we can hardly compare them.

>Unexpected weaknesses in ciphers designed by good cryptographers (Say
>Rivest, or Schneier)
>are very unlikely to appear.

Sorry. That is an insupportable statement (and of course unprovable). Being a crypto god does not prevent mistakes.

>Remember DES, after 25 years of analysis, is
>still only
>vulnerable to a brute force attack.

Also insupportable: DES may have a fast break right now, and if it does, would those who have it have told you? Yet you feel free to assume that there is no break. Good for you. Now prove it.

>I'd be willing to bet that RC-6 and
>TwoFish will withstand
>the same scrutiny.

But that is hardly science or even rational reasoning, is it?

Terry Ritter ritter@io.com http://www.io.com/~ritter/
Crypto Glossary http://www.io.com/~ritter/GLOSSARY.HTM

Subject: Re: Announce - ScramDisk v2.02h
Date: Fri, 9 Apr 1999 18:58:49 -0700
From: "Harvey Rook" <redrook@someyahoo.com>
Message-ID: <7emb5m$iq2@news.dns.microsoft.com>
References: <370e7e29.21686311@news.io.com>
Newsgroups: sci.crypt
Lines: 161

My comments withing...
Terry Ritter <ritter@io.com> wrote in message
news:370e7e29.21686311@news.io.com...
>
> On Fri, 9 Apr 1999 11:31:40 -0700, in
> <7elhhc$ic6@news.dns.microsoft.com>, in sci.crypt "Harvey Rook"
> <redrook@someyahoo.com> wrote:
>
> >>
> >This is incorrect. By Kerchhoff's maxim, you have to assume your attacker
> >has a copy of your deciphering machine. If he has a copy of your
deciphering
> >machine, the attacker can figure out the algorithm you use to select
> >ciphers.
>
> That is no more true than saying that the attacker can figure out the
> message key or the initialization vector. We assume the system has
> some reliable random source to make such values. Similar values
> select what ciphers to use in what order. And this should change
> frequently.
>

The reliable random source must be communicated to both the encryptor, and the decryptor. Because it's transmitted or shared, you must assume the attacker has intercepted it. Because the attacker has intercepted the it, the attacker knows what ciphers you are using.

>
> >Once he knows the algorithm used to select ciphers, super-encipherment
> >only doubles or triples the amount of time needed to brute force. You'd
> >be much better off adding an extra byte to your key.
>
> The issue is far beyond increasing the keyspace, since that would take
> a huge number of ciphers. The issue instead is about spreading
> messages among many different ciphers, thus forcing The Opponents to
> "keep up" by identifying, acquiring, attacking, and breaking a
> continuing flow of new ciphers. This forces The Opponents to invest
> far more, and breaking any of these gains them far less. Note the
> contrast to the current situation, where only a few main ciphers are
> used, so only those need be broken.
>

Good ciphers cannot be instantly generated. They must be individually scrutinized and tested. If you don't do that, you are setting yourself up for failure.

> >>[...]
> >> Yes. Even if each cipher used has known weaknesses, those may not be
> >> exploitable in the multi-ciphering case.
> >>
> >It's only harder by one or two orders of magnitude.
>
> Well, let's see: Suppose we have about 4K ciphers. By selecting 3 at
> random so we have about 4k**3 selections. This is somewhat larger
> than "two orders of magnitude."
>
> But, again, the issue is not really the increase in keyspace. The
> issue is that The Opponents have to increase their analysis budget by,
> say, a factor of 1k. And even when they are successful, they get at
> most 1 message out of 1k. And we reduce that to 1 in 1k*1k*1k by
> using 3 layers of cipher.
>

Why is this not the same as increasing key space? The algorithms
you use to choose the ciphers, must be transmitted to the decriptor.
Once the attacker intercepts this, he can deduce the ciphers used.

Which would you rather trust, one well analyzed cipher with 140 bits of
key, or 4096 ciphers that probably aren't well analyzed, 128 bits of key,
and some mechanism to flip between the ciphers.

>
> >Computers built 3 years
> >from now
> >will have enough power to compensate for this difference. Adding an extra
> >byte
> >to your key makes the problem harder by 4 to 8 orders of magnitude. This
is
> >much
> >harder to attack, yet it's simpler and cleaner to analyze. Simple clean
> >systems, are
> >much less likely to have holes.
>
> Using a layered structure is far, far simpler than trying to analyze a
> cipher to the degree of predicting that one cipher is stronger than
> another by n orders of magnitude. Neither value is known, so we can
> hardly compare them.
>
>
> >Unexpected weaknesses in ciphers designed by good cryptographers (Say
> >Rivest, or Schneier)
> >are very unlikely to appear.
>
> Sorry. That is an insupportable statement (and of course unprovable).
> Being a crypto god does not prevent mistakes.
>

Correct. However, because many ciphers have a similar structure.
An advance in discreet mathematics needed to crack one well
designed cipher would most likely apply to many ciphers. If you
have some kind of scheme that allowed you to generate thousands of
ciphers, then it is very likely that the weakness would apply to all.

I think it's very unlikely that some one could generate 4096 ciphers that
don't share a common weakness, and still properly analyzed all of them.
Even if they did, the selection of which ciphers to use, is part of the key.
Brute forcing an N bit key plus K bits of which-cipher materical takes
O(2^(N+K)) Bruit forcing an N+K bit key also takes O(2^(N+K+b))

See, for every cipher you add, there is an equivalent scheme with only
one cipher that has the same keys size.

I dare you to name a relevent security breakdown in the past
20 years that was not the result poor key managment, or poor protocol
design. We have good algorithms. Good security comes not form flipping
between the algorithms, but from using the algorithms properly.

> >Remember DES, after 25 years of analysis, is
> >still only
> >vulnerable to a brute force attack.
>
> Also insupportable: DES may have a fast break right now, and if it
> does, would those who have it have told you? Yet you feel free to
> assume that there is no break. Good for you. Now prove it.

In the same vein, prove to me that a significant percentage of your
4096 ciphers will be strong. Unless a cipher is widely scrutinized, by
many intelligent people, you must assume it is weak. A solid analysis
of 4096 ciphers would take decades. We need good security now.

It usually impossible to prove an impossibility. In the absense of
proof you have to bet on the evidence. We have evidence that a
few ciphers are strong. We also have evidence that most ciphers are
weak.

> >I'd be willing to bet that RC-6 and
> >TwoFish will withstand
> >the same scrutiny.
>
> But that is hardly science or even rational reasoning, is it?

>

The scientific prinipal is observe, hypothesize, experiment, repeat.
I can observe that one cipher looks strong. I can form hypothesis about
it's strengths and weaknesses. I can test these hypothesis, and I can repeat
until I trust.

With thousands of ciphers, I probably could not observe that they all
look strong. There are too many of them. The analysis would be
over whelming. How can I go from this state, to hypothesizing that
combining them would be secure?

Harv
RedRook At Some Yahoo Dot Com
Remove the Some to send email.

Subject: Re: Announce - ScramDisk v2.02h
Date: Sat, 10 Apr 1999 03:56:08 GMT
From: ritter@io.com (Terry Ritter)
Message-ID: <370ecbaa.2845710@news.io.com>
References: <7emb5m$iq2@news.dns.microsoft.com>
Newsgroups: sci.crypt
Lines: 252

On Fri, 9 Apr 1999 18:58:49 -0700, in
<7emb5m$iq2@news.dns.microsoft.com>, in sci.crypt "Harvey Rook"
<redrook@someyahoo.com> wrote:

>My comments withing...
>Terry Ritter <ritter@io.com> wrote in message
>news:370e7e29.21686311@news.io.com...
>>
>> On Fri, 9 Apr 1999 11:31:40 -0700, in
>> <7elhhc$ic6@news.dns.microsoft.com>, in sci.crypt "Harvey Rook"
>> <redrook@someyahoo.com> wrote:
>>
>> >>
>> >This is incorrect. By Kerchhoff's maxim, you have to assume your attacker
>> >has a copy of your deciphering machine. If he has a copy of your
>deciphering
>> >machine, the attacker can figure out the algorithm you use to select
>> >ciphers.
>>
>> That is no more true than saying that the attacker can figure out the
>> message key or the initialization vector. We assume the system has
>> some reliable random source to make such values. Similar values
>> select what ciphers to use in what order. And this should change
>> frequently.
>>
>
>The reliable random source must be communicated to both the encryptor,
>and the decryptor. Because it's transmitted or shared, you must assume
>the attacker has intercepted it. Because the attacker has intercepted the
>it, the attacker knows what ciphers you are using.

That is false. It is always necessary to transport keys in some way.
But whatever way this is done -- by courier, multiple channels, or
authenticated public key -- random message key value are known by both
ends. That keying value can be used to select ciphers.

In practice, I would expect background negotiation between the cipher
programs to occur with each message transmission, and produce a new
cipher set with each message or every couple of messages.

>> >Once he knows the algorithm used to select ciphers, super-encipherment
>> >only doubles or triples the amount of time needed to brute force. You'd
>> >be much better off adding an extra byte to your key.
>>
>> The issue is far beyond increasing the keyspace, since that would take
>> a huge number of ciphers. The issue instead is about spreading
>> messages among many different ciphers, thus forcing The Opponents to
>> "keep up" by identifying, acquiring, attacking, and breaking a
>> continuing flow of new ciphers. This forces The Opponents to invest
>> far more, and breaking any of these gains them far less. Note the
>> contrast to the current situation, where only a few main ciphers are
>> used, so only those need be broken.
>>
>
>Good ciphers cannot be instantly generated. They must be individually
>scrutinized and tested. If you don't do that, you are setting
>yourself up for failure.

Look around: My guess is that we could probably point to hundreds of
cipher designs which now exist. This is in an environment where there
is no recognition of the value of multiple ciphers, and no common
interface to handle plug-in cipher components. Were the environment
to change, I think we could easily see 500 ciphers in a few years, and
tens of thousands of ciphers in a couple of decades.

>> >>[...]

>> >> Yes. Even if each cipher used has known weaknesses, those may not be
>> >> exploitable in the multi-ciphering case.
>> >>
>> >It's only harder by one or two orders of magnitude.
>>
>> Well, let's see: Suppose we have about 4K ciphers. By selecting 3 at
>> random so we have about 4k**3 selections. This is somewhat larger
>> than "two orders of magnitude."
>>
>> But, again, the issue is not really the increase in keyspace. The
>> issue is that The Opponents have to increase their analysis budget by,
>> say, a factor of 1k. And even when they are successful, they get at
>> most 1 message out of 1k. And we reduce that to 1 in 1k*1k*1k by
>> using 3 layers of cipher.
>>
>
>Why is this not the same as increasing key space? The algorithms
>you use to choose the ciphers, must be transmitted to the decriptor.
>Once the attacker intercepts this, he can deduce the ciphers used.

Certainly, the cipher selection amounts to a sort of key, but this
would be a one-time message key or initialization vector, as opposed
to a repeatedly-used user key.

>Which would you rather trust, one well analyzed cipher with 140 bits of
>key, or 4096 ciphers that probably aren't well analyzed, 128 bits of key,
>and some mechanism to flip between the ciphers.

I take the 4096 ciphers, and I have explained why: First, the
multi-cipher situation forces any Opponent whatsoever to keep up in a
process which is vastly more expensive for them then for us. Next, we
divide our total traffic among many different ciphers. So even if an
Opponent breaks a cipher, the best they can hope for is 1/n of the
traffic. And in the multi-ciphering case, they don't even get that.

As to "trust," you should be aware that there *is* no way to prove
strength, to measure strength, to build strength or test strength.
Any "trust" of a cipher is a delusion only. For example, the best
cryptanalysis can do is find weakness, and this only helps if weakness
is actually found. If no weakness is found, cryptanalysis does not
testify to strength.

Personally, I think we are better off with user selection (e.g., "I
use what my friend uses," or "We bought a custom cipher subscription")
than any standard picked by a central authority.

>> >Computers built 3 years
>> >from now
>> >will have enough power to compensate for this difference. Adding an extra
>> >byte
>> >to your key makes the problem harder by 4 to 8 orders of magnitude. This
>is
>> >much
>> >harder to attack, yet it's simpler and cleaner to analyze. Simple clean
>> >systems, are
>> >much less likely to have holes.
>>
>> Using a layered structure is far, far simpler than trying to analyze a
>> cipher to the degree of predicting that one cipher is stronger than
>> another by n orders of magnitude. Neither value is known, so we can
>> hardly compare them.
>>
>>
>> >Unexpected weaknesses in ciphers designed by good cryptographers (Say
>> >Rivest, or Schneier)
>> >are very unlikely to appear.
>>
>> Sorry. That is an insupportable statement (and of course unprovable).
>> Being a crypto god does not prevent mistakes.
>>
>
>Correct. However, because many ciphers have a similar structure.
>An advance in discreet mathematics needed to crack one well
>designed cipher would most likely apply to many ciphers. If you
>have some kind of scheme that allowed you to generate thousands of
>ciphers, then it is very likely that the weakness would apply to all.

Certainly any cipher designer will have his bag of tricks. One might
expect that some of those tricks would be risky, but hardly all. And
the vast number of designers would be using many different approaches.
It seems extremely *un*likely that a single breakthrough would affect
them all. Indeed, the unexpected break would be far, far more likely
(and far, far more serious) if we have just one standard cipher.

>I think it's very unlikely that some one could generate 4096 ciphers that
>don't share a common weakness, and still properly analyzed all of them.
>Even if they did, the selection of which ciphers to use, is part of the key.
>Brute forcing an N bit key plus K bits of which-cipher materical takes
>O(2^(N+K)) Bruit forcing an N+K bit key also takes O(2^(N+K+b))
>
>See, for every cipher you add, there is an equivalent scheme with only
>one cipher that has the same keys size.

Only in the sense that the meta cipher has various components, but these are unknown until they are written. That meta cipher does not exist until nobody is writing new ciphers anymore.

>I dare you to name a relevent security breakdown in the past
>20 years that was not the result poor key managment, or poor protocol
>design.

DES keyspace.

>We have good algorithms. Good security comes not form flipping
>between the algorithms, but from using the algorithms properly.

In the present tense, it is certain that this rarely happens now. But the opportunity is available for the future.

>> >Remember DES, after 25 years of analysis, is
>> >still only
>> >vulnerable to a brute force attack.
>>
>> Also insupportable: DES may have a fast break right now, and if it
>> does, would those who have it have told you? Yet you feel free to
>> assume that there is no break. Good for you. Now prove it.
>
>In the same vein, prove to me that a significant percentage of your
>4096 ciphers will be strong. Unless a cipher is widely scrutinized, by
>many intelligent people, you must assume it is weak. A solid analysis
>of 4096 ciphers would take decades. We need good security now.

First of all, ciphers should be selected by users, and your so-called strength is up to them. If a cipher is found weak in some academic paper, the users may choose to disable that cipher from further use, or not. In contrast, if the one standard cipher is found weak, there is no real alternative, and no process for making that change.

Significant security is provided simply by partitioning the message space into many different ciphers; by having a continuing flow of new ciphers; and by making multi-ciphering an expected process. This is security added to any set of ciphers which may already exist.

Any widely scrutinized cipher could be used in this process, and would not be weakened by it. The many-cipher situation can only improve things, not weaken what is already there.

>It usually impossible to prove an impossibility. In the absense of
>proof you have to bet on the evidence. We have evidence that a
>few ciphers are strong. We also have evidence that most ciphers are
>weak.

In the absence of proof of strength, it is foolish to bet that a particular cipher is strong. The best bet is to use multiple ciphers at the same time, change ciphers often, and be prepared to disable particular ciphers at the first sign of trouble.

>> >I'd be willing to bet that RC-6 and
>> >TwoFish will withstand
>> >the same scrutiny.
>>
>> But that is hardly science or even rational reasoning, is it?
>>
>
>The scientific prinipal is observe, hypothesize, experiment, repeat.
>I can observe that one cipher looks strong. I can form hypothesis about
>it's strengths and weaknesses. I can test these hypothesis, and I can repeat
>until I trust.

No, you cannot test strength. Neither can anyone else. *That* is the problem.

>With thousands of ciphers, I probably could not observe that they all
>look strong. There are too many of them. The analysis would be
>over whelming. How can I go from this state, to hypothesizing that
>combining them would be secure?

By observing that any one cipher that you do accept can be part of stack. That would give you 2 changing ciphers, the ability to switch to another if you get bad news, and the knowledge that any hidden break to your favorite cipher is protected by the rest of the stack.

Terry Ritter ritter@io.com http://www.io.com/~ritter/
Crypto Glossary http://www.io.com/~ritter/GLOSSARY.HTM

Subject: Re: Announce - ScramDisk v2.02h
Date: Mon, 12 Apr 1999 10:31:55 GMT
From: ssimpson@hertreg.ac.uk
Message-ID: <7esi2rcct1@nnrp1.dejanews.com>
References: <7elhhc$ic6@news.dns.microsoft.com>

Newsgroups: sci.crypt
Lines: 41

In article <7elhhc$ic6@news.dns.microsoft.com>,
  "Harvey Rook" <redrook@someyahoo.com> wrote:
>
> Terry Ritter <ritter@io.com> wrote in message
> news:370e33ea.2675997@news.io.com...
> >
> > On 9 Apr 1999 11:10:56 GMT, in <7ekn80$kc1$1@korai.cygnus.co.uk>, in
> > sci.crypt aph@cygnus.remove.co.uk (Andrew Haley) wrote:
> >
> > >The answer is simple. Kerckhoff's maxim says that your attacker knows
> > >the cryptosystem you're using, but does not know the key. If you're
> > >using superencryption, your attacker knows which systems you're using.
> >
> > That's fine if you always use the same ciphers in the same order. But
> > if the ciphers are dynamically selected by keying, or just dynamically
> > selected frequently by communications under cipher, the attacker does
> > *not* know "which systems you're using." Kerckhoff's maxim does not
> > apply.
> >
> This is incorrect. By Kerchhoff's maxim, you have to assume your attacker
> has a copy of your deciphering machine. If he has a copy of your deciphering
> machine, the attacker can figure out the algorithm you use to select
> ciphers.

Make the cipher selection(s) dependant on additional key material?

> Once he knows the algorithm used to select ciphers, super-encipherment
> only doubles or triples the amount of time needed to brute force. You'd
> be much better off adding an extra byte to your key.

Depends on the method of selection, range of ciphers and number we are willing
to employ to encrypt one message doesn't it?

Sam Simpson
Comms Analyst
-- http://www.scramdisk.clara.net/ for ScramDisk hard-drive encryption &
Delphi Crypto Components. PGP Keys available at the same site.

----------== Posted via Deja News, The Discussion Network ==----------
http://www.dejanews.com/ Search, Read, Discuss, or Start Your Own

---

Subject: Re: Announce - ScramDisk v2.02h
Date: Fri, 09 Apr 1999 18:28:10 -0400
From: Boris Kazak <bkazak@worldnet.att.net>
Message-ID: <370E7EFA.7BB2@worldnet.att.net>
References: <370e33ea.2675997@news.io.com>
Newsgroups: sci.crypt
Lines: 19

Terry Ritter wrote:
> ....
> >Of course, your attacker must now analyze the compound cipher, which
> >is almost certainly harder to do than than attacking a single cipher.
>
> Yes. Even if each cipher used has known weaknesses, those may not be
> exploitable in the multi-ciphering case.
>
> ---
> Terry Ritter ritter@io.com http://www.io.com/~ritter/
> Crypto Glossary http://www.io.com/~ritter/GLOSSARY.HTM

  A practical suggestion: what if we would initialize BLOWFISH not
with the digits of 'pi', but with some other constant - e, sqrt(2),
sin(1)...? How many different ciphers can we generate from the same
base algorithm, and how big will be the additional effort to break
them all? Especially if dynamically selected...

   Best wishes BNK

Subject: Re: Announce - ScramDisk v2.02h
Date: Mon, 12 Apr 1999 10:26:58 GMT
From: ssimpson@hertreg.ac.uk
Message-ID: <7eshpd$c28$1@nnrp1.dejanews.com>
References: <7ekn80$kc1$1@korai.cygnus.co.uk>
Newsgroups: sci.crypt
Lines: 39

Why not make cipher selection for the compound cipher part of the key? The
first one or two bytes of the key could be used to select 3 (or whatever
number is desirable) ciphers out of a pool of equally good ciphers.

This sounds (imho) like a good idea - as long as there are no disastrously
weak combinations of ciphers.

Comments?

Sam Simpson Communications Analyst -- http://www.scramdisk.clara.net/ for
ScramDisk hard-drive encryption & Delphi Crypto Components. PGP Keys
available at the same site.

In article <7ekn80$kc1$1@korai.cygnus.co.uk>,
  aph@cygnus.remove.co.uk (Andrew Haley) wrote:
> [Newsgroups list trimmed]
>
> Lincoln Yeoh (lyeoh@pop.jaring.nospam.my) wrote:
>
> : I like the idea of superencryption too, and I don't know why so few
> : people seem to like it. So far I have not had a good answer to how
> : an attacker would know if he or she has succeeded.
>
> The answer is simple. Kerckhoff's maxim says that your attacker knows
> the cryptosystem you're using, but does not know the key. If you're
> using superencryption, your attacker knows which systems you're using.
>
> Of course, your attacker must now analyze the compound cipher, which
> is almost certainly harder to do than than attacking a single cipher.
>
> Andrew.

----------== Posted via Deja News, The Discussion Network ==----------
http://www.dejanews.com/ Search, Read, Discuss, or Start Your Own

---

Subject: Re: Announce - ScramDisk v2.02h
Date: Mon, 12 Apr 1999 15:30:03 -0600
From: jcoffin@taeus.com (Jerry Coffin)
Message-ID: <MPG.117bdb60312a13119899e2@news.rmi.net>
References: <7eshpd$c28$1@nnrp1.dejanews.com>
Newsgroups: sci.crypt
Lines: 16

In article <7eshpd$c28$1@nnrp1.dejanews.com>, ssimpson@hertreg.ac.uk
says...
> Why not make cipher selection for the compound cipher part of the key? The
> first one or two bytes of the key could be used to select 3 (or whatever
> number is desirable) ciphers out of a pool of equally good ciphers.
>
>
> This sounds (imho) like a good idea - as long as there are no disastrously
> weak combinations of ciphers.

This might optimize speed slightly, but I don't see it helping
security. If you're going to include code for a number of forms of
encryption, from a viewpoint of security, you might as well just
always use ALL the forms of encryption supported, and use the entire
key as a key instead some of it as a key and some to select the
method(s) of encryption to be used.

Subject: Re: Announce - ScramDisk v2.02h
Date: 22 Apr 1999 20:02:39 -0700
From: mskala@ansuz.sooke.bc.ca. (Matthew Skala)
Message-ID: <7fonsf$so3$1@ruby.ansuz.sooke.bc.ca>
References: <MPG.117bdb60312a13119899e2@news.rmi.net>
Newsgroups: sci.crypt
Lines: 18

In article <MPG.117bdb60312a13119899e2@news.rmi.net>,
Jerry Coffin <jcoffin@taeus.com> wrote:
>security. If you're going to include code for a number of forms of
>encryption, from a viewpoint of security, you might as well just
>always use ALL the forms of encryption supported, and use the entire

Speed. If you use all the ciphers to encrypt the whole file, it may take
you a very long time. But here's another idea: use some sort of
all-or-nothing scheme, so attackers have to attack the entire file at
once, and then encrypt say the first block with IDEA, the second block
with Blowfish, the third with 3DES, the fourth with SCOTT19U, and so on.
I'm not fully up on how well all-or-nothing schemes work, but it seems
like it should be possible to require the attackers to break all the
ciphers, while still not having to do all the ciphers on all the blocks.
--
Matthew Skala Ansuz BBS (250) 472-3169 http://www.islandnet.com/~mskala/

GOD HATES SPAM

Subject: Thought question: why do public ciphers use only simple ops like shift and
XOR?
Date: Fri, 16 Apr 1999 07:31:40 -0700
From: Sundial Services <info@sundialservices.com>
Message-ID: <371749CC.4779@sundialservices.com>
Newsgroups: sci.crypt
Lines: 18

When I look at most publicly-available cryptographic algorithms, I see that nearly all of them consist of round upon round of simple operations like: shift, exclusive-OR, and "bit-twiddling." Most of these ops are readily reversible.

About the only "original idea" I've seen, since reading discussions of older machines like SIGABA, is Terry Ritter's "Dynamic Substitution" patent. At least he is using a more complex transformation than 99.9% of the things I've seen ... since SIGABA ... and he's burying a lot more information than most designs do.

My question is, aside from possible requirements for constructing their ciphers in hardware, why do designers routinely limit themselves to these simple bitwise operators in designing ciphers? It seems to me as a layman that the older, more complex designs were also far more secure than what we have now, and that a computer program would have no particular difficulty implementing them. We are not building hardware devices; we are not limited to LFSR's.

Subject: Re: Thought question: why do public ciphers use only simple ops like shift and XOR?
Date: Fri, 16 Apr 1999 17:28:13 GMT
From: jsavard@tenMAPSONeerf.edmonton.ab.ca (John Savard)
Message-ID: <37176a30.4219613@news.prosurfr.com>
References: <371749CC.4779@sundialservices.com>
Newsgroups: sci.crypt
Lines: 111

Sundial Services <info@sundialservices.com> wrote, in part:

>When I look at most publicly-available cryptographic algorithms, I see
>that nearly all of them consist of round upon round of simple operations
>like: shift, exclusive-OR, and "bit-twiddling." Most of these ops are
>readily reversible.

Looking at this paragraph, and your title, my initial reaction was to say that you were wrong - block cipher designers do recognize the importance of nonlinearity, and thus in virtually every block cipher you will find an S-box.

>About the only "original idea" I've seen, since reading discussions of
>older machines like SIGABA, is Terry Ritter's "Dynamic Substitution"
>patent. At least he is using a more complex transformation than 99.9%
>of the things I've seen ... since SIGABA ... and he's burying a lot more
>information than most designs do.

Dynamic Substitution is a good idea, and an original one. And since I consider the SIGABA to be an admirable design, I started to warm to you at this point.

>My question is, aside from possible requirements for constructing their
>ciphers in hardware, why do designers routinely limit themselves to
>these simple bitwise operators in designing ciphers? It seems to me as
>a layman that the older, more complex designs were also far more secure
>than what we have now, and that a computer program would have no
>particular difficulty implementing them. We are not building hardware
>devices; we are not limited to LFSR's.

Now this is a question I've been asking myself.

But there are answers to it.

- For one thing, not everyone using cryptography is simply writing a program to encipher E-mail. If, for that application, it is trivial to just throw complexity at the problem to obtain security, then there's no money to be made from designing a cipher which is secure in that situation...anyone can do it. What about securely encrypting real-time digital video?

- Also, since there are many insecure cipher designs floating around, one can't just accept that a cipher is secure based on its designer's say-so. Instead, what gives real confidence in a cipher design is that it has been studied by experts who have failed to crack it, but who have come away from their attempts with an understanding of the source of the design's strengths.

But an academic researcher isn't going to take time studying a cipher that is so big and complicated that there is no hope of coming away with an impressive result - and so big and complicated that even trying to understand it would consume an enormous amount of time and effort.

Thus, designs that are intentionally limited - to one basic type of round, to one underlying principle - have an advantage over designs based on the principle that security is the only goal. They might be less intrinsically secure, but they have a better chance of being able to (appear to) _prove_ (indicate with some tendency to confidence) that they do have a certain level of security.

Although I do understand the rationale behind the "recieved wisdom", that doesn't mean I fully accept it. In practice, when using cryptography, security is what counts; and advances are being made both in the theory of cryptanalysis and in the speed and power of computer chips at a great rate.

Plus, the risk that one's adversary is a hacker of the future with a very powerful desktop computer seems much greater than the risk that one's adversary will be an accomplished cryptanalyst, able to exploit the most subtle flaws in an over-elaborate design.

Hence, I have played with designs that don't just use "simple operations". They do incorporate a lot from the designs of the real experts in the field, compared to which I am a mere amateur, but they go on from there to pile on a higher level of complication than seen in the well-known designs.

Take a look at my Quadibloc II and Quadibloc III designs, in

http://members.xoom.com/quadibloc/co040705.htm
http://members.xoom.com/quadibloc/co040705.htm

for example. I think they may address your concern - although they may not go far enough.

One thing I _very definitely_ don't want to do is to go around like certain posters on this NG, and claim that a cipher *must* be as complicated as these designs of mine in order to be secure. That simply isn't true.

And it is also true that a strong cipher isn't a guarantee of security; designing ciphers may be fun, but preventing data from leaking out the back door is hard work.

While I respect the knowledge and ability of the acknowledged experts in the field, where I think I part company with Bruce Schneier and others is in the following:

I believe it to be possible and useful to develop a design methodology - mainly involving the cutting and pasting of pieces from proven cipher designs - to enable a reasonably qualified person who, however, falls short of being a full-fleged cryptographer, to design his own block cipher, and thereby obtain additional and significant benefits in resistance to cryptanalytic attack by having an unknown and unique algorithm.

I don't deny that there are pitfalls looming in such an approach; if something is left out of the methodology, or if it isn't conscientiously used, people could easily wind up using weak designs and having a false sense of security. I just think the problems can be addressed, and the potential benefits are worth the attempt.

John Savard (teneerf is spelled backwards)
http://members.xoom.com/quadibloc/index.html

Subject: Re: Thought question: why do public ciphers use only simple ops like shift and XOR?
Date: Fri, 16 Apr 1999 20:20:24 GMT
From: ritter@io.com (Terry Ritter)
Message-ID: <37179b67.12809750@news.io.com>
References: <37176a30.4219613@news.prosurfr.com>
Newsgroups: sci.crypt
Lines: 129

On Fri, 16 Apr 1999 17:28:13 GMT, in
<37176a30.4219613@news.prosurfr.com>, in sci.crypt
jsavard@tenMAPSONeerf.edmonton.ab.ca (John Savard) wrote:

>[...]
>- Also, since there are many insecure cipher designs floating around, one
>can't just accept that a cipher is secure based on its designer's say-so.
>Instead, what gives real confidence in a cipher design is that it has been
>studied by experts who have failed to crack it, but who have come away from
>their attempts with an understanding of the source of the design's
>strengths.

I dispute this. This is essentially what Schneier would have us
believe, and it is false.

The truth is that we *never* know the "real" strength of a cipher. No
matter how much review or cryptanalysis a cipher gets, we only have
the latest "upper bound" for strength. The lower bound is zero: Any
cipher can fail at any time.

Since we have only an upper bound for the strength of any cipher, any
confidence we may have is no more than our own delusion. We wish and
hope for cipher strength, and -- absent a specific proof otherwise --
we gradually come to believe in it. But that does not make it true.

We would like to think that the more we use a cipher, the more
confidence we can have in it. We *can* build confidence in a
ciphering program, as to whether or not it crashes and so on. But
since our Opponents do not tell us of their success, we do not know
that our cipher was successful at hiding data. And we cannot have
confidence in a result without knowing what that result is.

>[...]
>But an academic researcher isn't going to take time studying a cipher that
>is so big and complicated that there is no hope of coming away with an

>impressive result - and so big and complicated that even trying to
>understand it would consume an enormous amount of time and effort.

It is always nice to find something important which is easy to do.
That would be the academic equivalent of "Make Easy Money Now."

It may be unfortunate for academic cryptographers that a wide variety of new techniques are pioneered by non-academics. But those techniques exist nevertheless, and to the extent that academics do not investigate them, those academics are not up with the state of the art.

It is not, frankly, the role of the innovator to educate the academics, or even to serve technology to them on a silver platter. In the end, academic reputation comes from reality, and the reality is that many crypto academics avoid anything new which does not have an academic source. The consequence is that they simply do not have the background to judge really new designs.

>Thus, designs that are intentionally limited - to one basic type of round,
>to one underlying principle - have an advantage over designs based on the
>principle that security is the only goal. They might be less intrinsically
>secure, but they have a better chance of being able to (appear to) _prove_
>(indicate with some tendency to confidence) that they do have a certain
>level of security.

Upon encountering a new design, anyone may choose to simplify that design and then report results from that simplification. This is done all the time. It is not necessary for an innovator to make a simplified design for this purpose.

On the other hand, I have been pioneering the use of scalable technology which, presumably, can be scaled down to a level which can be investigated experimentally. The last I heard, experimentation was still considered a rational basis for the understanding of reality. Indeed, one might argue that in the absence of theoretical strength for *any* cipher, experimentation is about all we have. But note how little of it we see.

>[...]
>Plus, the risk that one's adversary is a hacker of the future with a very
>powerful desktop computer seems much greater than the risk that one's
>adversary will be an accomplished cryptanalyst, able to exploit the most
>subtle flaws in an over-elaborate design.

But we don't know our Opponents! If we have to estimate their capabilities, I think we are necessarily forced into assuming that they are more experienced, better equipped, have more time, are better motivated, and -- yes -- are even smarter than we are. There is ample opportunity for them to exploit attacks of which we have no inkling at all.

>[...]
>While I respect the knowledge and ability of the acknowledged experts in
>the field, where I think I part company with Bruce Schneier and others is
>in the following:
>
>I believe it to be possible and useful to develop a design methodology -
>mainly involving the cutting and pasting of pieces from proven cipher
>designs - to enable a reasonably qualified person who, however, falls short
>of being a full-fleged cryptographer, to design his own block cipher, and
>thereby obtain additional and significant benefits in resistance to
>cryptanalytic attack by having an unknown and unique algorithm.

And in this way we can have hundreds or thousands of different ciphers, with more on the way all the time. That means that we can divide the worth of our information into many different ciphers, so that if any one fails, only a fraction of messages are exposed. It also means that *any* Opponent must keep up with new ciphers and analyze and possibly break each, then design a program, or build new hardware to exploit it. We can make good new ciphers cheaper than they can possibly be broken. The result is that our Opponents must invest far more to get far less, and this advantage does not depend upon the delusion of strength which is all that cryptanalysis can provide.

>I don't deny that there are pitfalls looming in such an approach; if
>something is left out of the methodology, or if it isn't conscientiously
>used, people could easily wind up using weak designs and having a false
>sense of security. I just think the problems can be addressed, and the
>potential benefits are worth the attempt.

Neat.

Terry Ritter ritter@io.com http://www.io.com/~ritter/
Crypto Glossary http://www.io.com/~ritter/GLOSSARY.HTM

Subject: Re: Thought question: why do public ciphers use only simple ops like shift

and XOR?
Date: Fri, 16 Apr 1999 14:06:57 -0700
From: "Steven Alexander" <steve@cell2000.net>
Message-ID: <jKNR2.591$%L2.8044@news6.ispnews.com>
References: <37179b67.12809750@news.io.com>
Newsgroups: sci.crypt
Lines: 32

>>- Also, since there are many insecure cipher designs floating around, one
>>can't just accept that a cipher is secure based on its designer's say-so.
>>Instead, what gives real confidence in a cipher design is that it has been
>>studied by experts who have failed to crack it, but who have come away
from
>>their attempts with an understanding of the source of the design's
>>strengths.
>
>I dispute this. This is essentially what Schneier would have us
>believe, and it is false.
>
>The truth is that we *never* know the "real" strength of a cipher. No.....

I don't think that you understand the point that Schneier and others have
made. If I(a nobody) create a new cryptosystem tommorrow, nobody will have
any confidence in it. But, If I learn to break the ciphers of others and
use my experience to create a new cipher that others cannot break it will be
listened to because I am known to be knowledgeable in how ciphers work.
But, it will still not be trusted. Only after many people have analyzed and
failed to break my cipher will people say..."his cipher has held up to
five(ten) years of cryptanalysis by very knowledgeable cryptanalysts. We
can assume with an adequate level of confidence that the cipher will protect
our information." However, it is still realized that at any time someone
can invent a new cryptanalytic attack and my cipher will be rendered
useless. Schneier and others have acknowledged that any cipher can be
broken at any time.

my $.02...-steven

Subject: Re: Thought question: why do public ciphers use only simple ops like shift
and XOR?
Date: Fri, 16 Apr 1999 22:32:57 GMT
From: ritter@io.com (Terry Ritter)
Message-ID: <3717ba72.20758328@news.io.com>
References: <jKNR2.591$%L2.8044@news6.ispnews.com>
Newsgroups: sci.crypt
Lines: 87

On Fri, 16 Apr 1999 14:06:57 -0700, in
<jKNR2.591$%L2.8044@news6.ispnews.com>, in sci.crypt "Steven
Alexander" <steve@cell2000.net> wrote:

>>[...]
>>I dispute this. This is essentially what Schneier would have us
>>believe, and it is false.
>>
>>The truth is that we *never* know the "real" strength of a cipher. No.....
>
>I don't think that you understand the point that Schneier and others have
>made.
>If I(a nobody) create a new cryptosystem tommorrow, nobody will have
>any confidence in it.

This is seriously disturbing: The issue is not who makes a thing, but
instead what the thing actually is. Deliberately judging a design in
the context of who made it is actually anti-scientific, and should be
widely denounced as the superstition it is.

>But, If I learn to break the ciphers of others and
>use my experience to create a new cipher that others cannot break it will be
>listened to because I am known to be knowledgeable in how ciphers work.

Nonsense. Knowing how to break some ciphers does not mean that you
know how ciphers work. That idea *is* the point "that Schneier and
others have made" and it is a fantasy. It is especially fantastic
when ciphers use technology which academics have ignored. But in any
case, without a LOWER bound on strength, academics REALLY do not even
know that ciphers work *at* *all*, let alone how.

>But, it will still not be trusted. Only after many people have analyzed and
>failed to break my cipher will people say...CRYPHTML.HTM"his cipher has held up to
>five(ten) years of cryptanalysis by very knowledgeable cryptanalysts.

Nonsense. There is no such conclusion. Ciphers do not ripen like
cheese.

We first of all do not know how many attacks were made (if any), nor
how much effort was placed into them. Attacks made by experienced,
well-paid, well-motivated teams with all the equipment they need are
quite different from those of single individuals working at a desk at

night and coming up with a new mathematical equation. Not finding an equation does not mean some team has not had success.

We only know what success is reported in the academic literature. Unfortunately, when we use a cipher, we are very rarely concerned whether academics can break our cipher or not. We are instead concerned about "bad guys," and they don't tell us when they have been successful.

So this delay -- supposedly for gaining confidence -- in reality tells us nothing at all about the strength of the cipher.

>We
>can assume with an adequate level of confidence that the cipher will protect
>our information." However, it is still realized that at any time someone
>can invent a new cryptanalytic attack and my cipher will be rendered
>useless. Schneier and others have acknowledged that any cipher can be
>broken at any time.

As I recall, Schneier and others claim that cryptanalysis is how we know the strength of a cipher. It is not. Cryptanalysis can only show weakness, only that when it is successful, and even then it only gives us the latest upper bound.

But the main problem is not knowing the strength of *new* ciphers, but rather knowing the strength of *old* ciphers: we are actually using the old ciphers. When ciphers have been in long use there is a delusion that we know their strength and can use them as a benchmark against new ciphers. Absent a non-zero LOWER bound on strength, this is false on both counts.

As I recall, in his comments on AES, Schneier has said that simply finding a cryptanalytic weakness in one of the designs would be sufficient to remove it from competition, even if the weakness was impractical. He would thus have us believe that the lack of information about weakness in one cipher is superior to information of impractical weakness in another cipher. I disagree.

Terry Ritter ritter@io.com http://www.io.com/~ritter/
Crypto Glossary http://www.io.com/~ritter/GLOSSARY.HTM

Subject: Re: Thought question: why do public ciphers use only simple ops like shift and XOR?
Date: Fri, 16 Apr 1999 15:41:12 -0700
From: "Steven Alexander" <steve@cell2000.net>
Message-ID: <X6PR2.1145$5E.10730@news7.ispnews.com>
References: <3717ba72.20758328@news.io.com>
Newsgroups: sci.crypt
Lines: 18

I think the point that Schneier and others have made, which I personally agree with, is that no cipher is "secure". We can however put more trust into an algorithm that has undergone more cryptanalysis and has been tested against the newest cryptanalytic techniques because we know what will not break the cipher. I personally would not trust any algorithm that I and other motivated people had not tested. I also think that understanding how to break ciphers gives a better knowledge of how to build ciphers because you know what can break them. This is why some of the best security experts are hackers...they know how to get in. You cannot prevent your computer from being hacked if you do not know what means someone will use to break in. It would be like building large stone walls around a military base and not expecting someone to fly over and drop a bomb...if you don't know that airplanes and bombs can destroy your base as well as ground troops...you've already lost.

-steven

Subject: Re: Thought question: why do public ciphers use only simple ops like shift and XOR?
Date: Fri, 16 Apr 1999 23:53:14 GMT
From: ritter@io.com (Terry Ritter)
Message-ID: <3717cd62.25607206@news.io.com>
References: <X6PR2.1145$5E.10730@news7.ispnews.com>
Newsgroups: sci.crypt
Lines: 61

On Fri, 16 Apr 1999 15:41:12 -0700, in <X6PR2.1145$5E.10730@news7.ispnews.com>, in sci.crypt "Steven Alexander" <steve@cell2000.net> wrote:

>I think the point that Schneier and others have made, which I personally
>agree with, is that no cipher is "secure".

I think you are being selective in stating "the" point Schneier has made. While he may have conceded that no cipher is secure after long discussion, his point often is that cryptanalysis is necessary to know the strength of a cipher. Of course, the fact that he sells such services would have nothing to do with it.

>We can however put more trust
>into an algorithm that has undergone more cryptanalysis and has been tested
>against the newest cryptanalytic techniques because we know what will not
>break the cipher.

Nope. Simply because "we" cannot break it does not mean that others cannot break it. We are not confronting our clones: our Opponents know more than we do, and are probably smarter as well.

>I personally would not trust any algorithm that I and
>other motivated people had not tested.

But there *is* no test for strength.

>I also think that understanding how
>to break ciphers

But there is no one way, nor any fixed set of ways, which are "how to break ciphers." No matter how much you "understand," there is more to know. That is the problem.

>gives a better knowledge of how to build ciphers because
>you know what can break them.

One proper role for cryptanalysis is to support the design of ciphers.

>This is why some of the best security experts
>are hackers...they know how to get in. You cannot prevent your computer
>from being hacked if you do not know what means someone will use to break
>in. It would be like building large stone walls around a military base and
>not expecting someone to fly over and drop a bomb...if you don't know that
>airplanes and bombs can destroy your base as well as ground troops...you've
>already lost.

Then you are lost. Neither you nor anybody else can predict every possible way to attack a cipher or a base.

Terry Ritter ritter@io.com http://www.io.com/~ritter/
Crypto Glossary http://www.io.com/~ritter/GLOSSARY.HTM

Subject: Re: Thought question: why do public ciphers use only simple ops like shift and XOR?
Date: Fri, 16 Apr 1999 17:05:05 -0700
From: "Steven Alexander" <steve@cell2000.net>
Message-ID: <xlQR2.1311$5E.12276@news7.ispnews.com>
References: <3717cd62.25607206@news.io.com>
Newsgroups: sci.crypt
Lines: 19

What exactly is your suggestion for the creation of a cipher in which we can place our trust? The best we can do at any one point is to create a cipher that is secure against the attacks that we know of . If we do not know of many attacks this will not entail much. If we have a group of the best cryptanalysts who analyze a cipher and find no vulnerabilities, this does not mean that any vulnerabilities do not exist...it only means that those that we know of...and variations thereof do not exist in that cipher. This gives us a degree of trust in the cipher. In RSA for example, we believe that the only way to break the cipher is to factor n. If I find a new way to factor n in just a couple of minutes on your typical PC the cipher is broken. However, the odds that someone will invent a way to factor that is so phenomenally better is very unlikely. If I try to build a cipher and do not understand cryptanalysis I will not ahve any idea how to protect my cipher. If you have a better way to design ciphers, please share.

-steven

Subject: Re: Thought question: why do public ciphers use only simple ops like shift and XOR?
Date: Sat, 17 Apr 1999 04:39:12 GMT
From: ritter@io.com (Terry Ritter)
Message-ID: <3718105d.5227815@news.io.com>
References: <xlQR2.1311$5E.12276@news7.ispnews.com>
Newsgroups: sci.crypt
Lines: 58

On Fri, 16 Apr 1999 17:05:05 -0700, in <xlQR2.1311$5E.12276@news7.ispnews.com>, in sci.crypt "Steven Alexander" <steve@cell2000.net> wrote:

>What exactly is your suggestion for the creation of a cipher in which we can
>place our trust?

Absent a theory or overall test of strength, there can be no trust in

a cipher. All the trust one can have is delusion.

>The best we can do at any one point is to create a cipher
>that is secure against the attacks that we know of . If we do not know of
>many attacks this will not entail much. If we have a group of the best
>cryptanalysts who analyze a cipher and find no vulnerabilities, this does
>not mean that any vulnerabilities do not exist...it only means that those
>that we know of...and variations thereof do not exist in that cipher.

Exactly.

>This
>gives us a degree of trust in the cipher.

What most people want is a strong cipher. Absent evidence of strength
there is no basis for such trust.

>In RSA for example, we believe
>that the only way to break the cipher is to factor n. If I find a new way
>to factor n in just a couple of minutes on your typical PC the cipher is
>broken. However, the odds that someone will invent a way to factor that is
>so phenomenally better is very unlikely.

This is a disturbingly-unwarranted statement: Nobody has any idea
what the true odds are, so we cannot infer that they are good or bad.

>If I try to build a cipher and do
>not understand cryptanalysis I will not ahve any idea how to protect my
>cipher. If you have a better way to design ciphers, please share.

Actually, I think there are better ways. For one thing we can use
very simple constructs with few types of component, each of which can
be fully understood for what it does. For another we can design
scalable ciphers that can be scaled down to experimental size.

However, the real issue is that while supposedly everyone knows that
any cipher can be weak, there has been essentially no attention given
to protocols which deal with this problem.

Terry Ritter ritter@io.com http://www.io.com/~ritter/
Crypto Glossary http://www.io.com/~ritter/GLOSSARY.HTM

Subject: Re: Thought question: why do public ciphers use only simple ops like shift
and XOR?
Date: Sun, 18 Apr 1999 22:09:10 GMT
From: jsavard@tenMAPSONeerf.edmonton.ab.ca (John Savard)
Message-ID: <371a5737.341699@news.prosurfr.com>
References: <xlQR2.1311$5E.12276@news7.ispnews.com>
Newsgroups: sci.crypt
Lines: 18

"Steven Alexander" <steve@cell2000.net> wrote, in part:

>If I try to build a cipher and do
>not understand cryptanalysis I will not ahve any idea how to protect my
>cipher. If you have a better way to design ciphers, please share.

You are right that avoiding known weaknesses is important, and
understanding cryptanalysis is important.

However, I think that there is a "better way to design ciphers" than to
place too much faith in the _present_ knowledge of cryptanalysis. A cipher
should be designed conservatively: not just in the sense of having a few
extra rounds, but in the sense of having extra complexities in its design
far beyond those needed (nonlinear S-boxes, irregularities in the key
schedule) to frustrate _known_ methods of attack.

John Savard (teenerf<-)
http://members.xoom.com/quadibloc/index.html

Subject: Re: Thought question: why do public ciphers use only simple ops like shift
and XOR?
Date: Sun, 18 Apr 1999 23:55:28 -0600
From: jgfunj@vgrknf.arg (wtshaw)
Message-ID: <jgfunj-1804992355290001@dial-243-098.itexas.net>
References: <371a5737.341699@news.prosurfr.com>
Newsgroups: sci.crypt
Lines: 26

In article <371a5737.341699@news.prosurfr.com>,
jsavard@tenMAPSONeerf.edmonton.ab.ca (John Savard) wrote:

> "Steven Alexander" <steve@cell2000.net> wrote, in part:
>
> >If I try to build a cipher and do

> >not understand cryptanalysis I will not ahve any idea how to protect my
> >cipher. If you have a better way to design ciphers, please share.
>
> You are right that avoiding known weaknesses is important, and
> understanding cryptanalysis is important.
>
> However, I think that there is a "better way to design ciphers" than to
> place too much faith in the _present_ knowledge of cryptanalysis. A cipher
> should be designed conservatively: not just in the sense of having a few
> extra rounds, but in the sense of having extra complexities in its design
> _far beyond_ those needed (nonlinear S-boxes, irregularities in the key
> schedule) to frustrate _known_ methods of attack.
>
A good trick is to telescope complexities into new primatives if you can.
Multiple layers of appropriate complexity do work, but the cost is
diversified in several directions.
--
A new random permutation generator: You put X windoze
machines in a room, merely start them up, and record the
order in which they eventually crash on their own.

Subject: Re: Thought question: why do public ciphers use only simple ops like shift
and XOR?
Date: Sat, 17 Apr 1999 02:22:42 GMT
From: fqkhuo@gmrvavvrcd.fl (ybizmt)
Message-ID: <slrn7hfs2u.cc.fqkhuo@tpep.nofsozwovh.yq>
References: <3717cd62.25607206@news.io.com>
Newsgroups: sci.crypt
Lines: 9

On Fri, 16 Apr 1999 23:53:14 GMT, Terry Ritter <ritter@io.com> wrote:
> *I* think you are being selective in stating "the" point Schneier has
> made. While he may have conceded that no cipher is secure after long
> discussion, his point often is that cryptanalysis is necessary to know
> the strength of a cipher. Of course, the fact that he sells such
> services would have nothing to do with it.

Refresh my memory. What do you sell?

Subject: Re: Thought question: why do public ciphers use only simple ops like shift
and XOR?
Date: Sat, 17 Apr 1999 04:39:19 GMT
From: ritter@io.com (Terry Ritter)
Message-ID: <37181072.5248874@news.io.com>
References: <slrn7hfs2u.cc.fqkhuo@tpep.nofsozwovh.yq>
Newsgroups: sci.crypt
Lines: 26

On Sat, 17 Apr 1999 02:22:42 GMT, in
<slrn7hfs2u.cc.fqkhuo@tpep.nofsozwovh.yq>, in sci.crypt
fqkhuo@gmrvavvrcd.fl (ybizmt) wrote:

>On Fri, 16 Apr 1999 23:53:14 GMT, Terry Ritter <ritter@io.com> wrote:
>> *I* think you are being selective in stating "the" point Schneier has
>> made. While he may have conceded that no cipher is secure after long
>> discussion, his point often is that cryptanalysis is necessary to know
>> the strength of a cipher. Of course, the fact that he sells such
>> services would have nothing to do with it.
>
>Refresh my memory. What do you sell?

Just the truth, lately.

I just find it an interesting coincidence when people promote errors
in reasoning which just happen to benefit their business.

On the other hand, promoting truths which also happen to benefit one's
business seems not nearly as disturbing.

Terry Ritter ritter@io.com http://www.io.com/~ritter/
Crypto Glossary http://www.io.com/~ritter/GLOSSARY.HTM

Subject: Re: Thought question: why do public ciphers use only simple ops like shift
and XOR?
Date: 18 Apr 99 01:49:42 GMT
From: jsavard@ecn.ab.ca ()
Message-ID: <37193a36.0@ecn.ab.ca>
References: <3717ba72.20758328@news.io.com>
Newsgroups: sci.crypt
Lines: 15

Terry Ritter (ritter@io.com) wrote:
: This is seriously disturbing: The issue is not who makes a thing, but
: instead what the thing actually is. Deliberately judging a design in
: the context of who made it is actually anti-scientific, and should be
: widely denounced as the superstition it is.

That's true *if* judging a cipher that way is used as a substitute for

actual analytical study of the cipher itself by a competent individual.
Where the services of an expert are not available, or there is
insufficient time to fully evaluate all candidate ciphers for an
application, choosing a cipher from a respected source is not
"superstition", and it is the kind of choice people make all the time:
i.e., when shopping for a new computer.

John Savard

Subject: Re: Thought question: why do public ciphers use only simple ops like shift
and XOR?
Date: Tue, 20 Apr 1999 22:03:24 GMT
From: ritter@io.com (Terry Ritter)
Message-ID: <371cf99f.7573878@news.io.com>
References: <37193a36.0@ecn.ab.ca>
Newsgroups: sci.crypt
Lines: 33

On 18 Apr 99 01:49:42 GMT, in <37193a36.0@ecn.ab.ca>, in sci.crypt
jsavard@ecn.ab.ca () wrote:

>Terry Ritter (ritter@io.com) wrote:
>: This is seriously disturbing: The issue is not who makes a thing, but
>: instead what the thing actually is. Deliberately judging a design in
>: the context of who made it is actually anti-scientific, and should be
>: widely denounced as the superstition it is.
>
>That's true *if* judging a cipher that way is used as a substitute for
>actual analytical study of the cipher itself by a competent individual.
>Where the services of an expert are not available, or there is
>insufficient time to fully evaluate all candidate ciphers for an
>application, choosing a cipher from a respected source is not
>"superstition", and it is the kind of choice people make all the time:
>i.e., when shopping for a new computer.

Is shopping for a cipher like shopping for a new computer? Yes, I
think so, but this situation is not a technical discussion between
people of expertise but, rather, ordinary users who really have no
choice but to rely upon promotion and rumor.

When experts themselves cannot fully characterize the strength of a
system specifically designed to produce strength, we know we are in
trouble. It's just that this is the way it's always been, and most of
us forgot what it means. It does not mean that we must rely upon the
same promotion and rumor as ordinary users.

Terry Ritter ritter@io.com http://www.io.com/~ritter/
Crypto Glossary http://www.io.com/~ritter/GLOSSARY.HTM

Subject: Re: Thought question: why do public ciphers use only simple ops like shift
and XOR?
Date: Tue, 20 Apr 1999 22:50:19 -0600
From: jgfunj@vgrknf.arg (wtshaw)
Message-ID: <jgfunj-2004992250200001@dial-243-073.itexas.net>
References: <371cf99f.7573878@news.io.com>
Newsgroups: sci.crypt
Lines: 14

In article <371cf99f.7573878@news.io.com>, ritter@io.com (Terry Ritter) wrote:
>
> Is shopping for a cipher like shopping for a new computer? Yes, I
> think so, but this situation is not a technical discussion between
> people of expertise but, rather, ordinary users who really have no
> choice but to rely upon promotion and rumor.
>
I wonder if the FTC has a role in determining if claims are reasonable.
They would have to yield to NSA for expertise? Perhaps we can try to
shift burden directly to government to prove strength, therefore making
them show their hand.
--
Life's battles do not always go to the stronger of faster man...
But, sooner or later always go to the fellow who thinks he can.

Subject: Re: Thought question: why do public ciphers use only simple ops like shift
and XOR?
Date: Sat, 17 Apr 1999 00:28:46 -0600
From: jgfunj@vgrknf.arg (wtshaw)
Message-ID: <jgfunj-1704990028460001@dial-243-079.itexas.net>
References: <jKNR2.591$%L2.8044@news6.ispnews.com>
Newsgroups: sci.crypt
Lines: 35

In article <jKNR2.591$%L2.8044@news6.ispnews.com>, "Steven Alexander"
<steve@cell2000.net> wrote:
>
> I don't think that you understand the point that Schneier and others have
> made. If I(a nobody) create a new cryptosystem tommorrow, nobody will have
> any confidence in it. But, If I learn to break the ciphers of others and

> use my experience to create a new cipher that others cannot break it will be
> listened to because I am known to be knowledgeable in how ciphers work.
> But, it will still not be trusted. Only after many people have analyzed and
> failed to break my cipher will people say...CRYPHTML.HTM"his cipher has held up to
> five(ten) years of cryptanalysis by very knowledgeable cryptanalysts. We
> can assume with an adequate level of confidence that the cipher will protect
> our information." However, it is still realized that at any time someone
> can invent a new cryptanalytic attack and my cipher will be rendered
> useless. Schneier and others have acknowledged that any cipher can be
> broken at any time.
>

You are still living in the same furrow. What matters is whether a cipher is good, and it will be so regardless of confidence bestowed by some select group fixated on a remarkedly few, perhaps some wrong, design criteria.

Converting unearned trust into acceptability can make a poor cipher pass for more than it is, and cause a great cipher to not get any attention. Your statement unfortunately often is a self-fulfilling prophesy that certain ciphers of a narrow nature will be given undue attention and consequently are more likely to get accepted. I would rather that people learn to not follow the leader so closely; it's a big world out there worth exploring cryptologically.

One thing I do like about the AES process is that there was some diversity, not enough, but some. Unfortunately, the target was more influenced by those who were creatures of the furrow.
--
Too much of a good thing can be much worse than none.

Subject: Re: Thought question: why do public ciphers use only simple ops like shift and XOR?
Date: Sat, 17 Apr 1999 20:36:40 -0400
From: "Trevor Jackson, III" <fullmoon@aspi.net>
Message-ID: <37192918.13924DDE@aspi.net>
References: <jKNR2.591$%L2.8044@news6.ispnews.com>
Newsgroups: sci.crypt
Lines: 50

Steven Alexander wrote:
>
> >>- Also, since there are many insecure cipher designs floating around, one
> >>can't just accept that a cipher is secure based on its designer's say-so.
> >>Instead, what gives real confidence in a cipher design is that it has been
> >>studied by experts who have failed to crack it, but who have come away
> from
> >>their attempts with an understanding of the source of the design's
> >>strengths.
> >
> >I dispute this. This is essentially what Schneier would have us
> >believe, and it is false.
> >
> >The truth is that we *never* know the "real" strength of a cipher. No.....
>
> I don't think that you understand the point that Schneier and others have
> made. If I(a nobody) create a new cryptosystem tommorrow, nobody will have
> any confidence in it. But, If I learn to break the ciphers of others and
> use my experience to create a new cipher that others cannot break it will be
> listened to because I am known to be knowledgeable in how ciphers work.
> But, it will still not be trusted. Only after many people have analyzed and
> failed to break my cipher will people say...CRYPHTML.HTM"his cipher has held up to
> five(ten) years of cryptanalysis by very knowledgeable cryptanalysts. We
> can assume with an adequate level of confidence that the cipher will protect
> our information." However, it is still realized that at any time someone
> can invent a new cryptanalytic attack and my cipher will be rendered
> useless. Schneier and others have acknowledged that any cipher can be
> broken at any time.
>

There's a name for this attitude. It's called the Aristotelean Fallacy -- the appeal to authority. It dominated science for centuries, and science suffered for it.

But even granting that I would prefer to purchase cryptographic products from a professional rather than an amateur, all this changes is the unit of measure. Instead of measuring the quality of the product we'll end up measuring the quality of the author. Now it's hard enough to define a unit of measure for ciphers. Imagine defining the unit of measure for cipher designers.

The fact that the best (only) standard we have for judging ciphers and their implementations is that of Brand Names indicates just how young/volatile/immature the field is. We've got good mathematical tools and good software engineering tools, but the toolbox for the crypto designer is mostly defined in the negative; by the toolbox of the crypto analyst.

When we have crypto-engineering standards similar to civil-engineering standards, we'll have a mature science (and very little excitement :-).

Subject: Re: Thought question: why do public ciphers use only simple ops like shift and XOR?

Date: Sun, 18 Apr 1999 00:28:12 -0600
From: jgfunj@vgrknf.arg (wtshaw)
Message-ID: <jgfunj-1804990028130001@dial-243-094.itexas.net>
References: <37192918.13924DDE@aspi.net>
Newsgroups: sci.crypt
Lines: 35

In article <37192918.13924DDE@aspi.net>, "Trevor Jackson, III"
<fullmoon@aspi.net> wrote:
>
> There's a name for this attitude. It's called the Aristotelean Fallacy
> -- the appeal to authority. It dominated science for centuries, and
> science suffered for it.
>
> But even granting that I would prefer to purchase cryptographic products
> from a professional rather than an amateur, all this changes is the unit
> of measure. Instead of measuring the quality of the product we'll end
> up measuring the quality of the author. Now it's hard enough to define
> a unit of measure for ciphers. Imagine defining the unit of measure for
> cipher designers.

The most professional cryptographic designers, the opponents, in the world
have offered of late...dung.
>
> The fact that the best (only) standard we have for judging ciphers and
> their implementations is that of Brand Names indicates just how
> young/volatile/immature the field is. We've got good mathematical tools
> and good software engineering tools, but the toolbox for the crypto
> designer is mostly defined in the negative; by the toolbox of the crypto
> analyst.

So they would have you believe.
>
> When we have crypto-engineering standards similar to civil-engineering
> standards, we'll have a mature science (and very little excitement :-).

Over standardization, regulation, formalizaton, and authoritarization has
killed many a good field. Maturation is not the enemy of creative, but
wheeler-dealer, power-sponges, who imagine that everyone else must follow
their lead, are.
--
Too much of a good thing can be much worse than none.

Subject: Re: Thought question: why do public ciphers use only simple ops like shift
and XOR?
Date: 21 Apr 1999 15:43:53 -0400
From: budney@peregrine.maya.com (Leonard R. Budney)
Message-ID: <m3d80xwyh2.fsf@peregrine.maya.com>
References: <37192918.13924DDE@aspi.net>
Newsgroups: sci.crypt
Lines: 66

"Trevor Jackson, III" <fullmoon@aspi.net> writes:

> Steven Alexander wrote:
> > If I learn to break the ciphers of others and use my experience to
> > create a new cipher that others cannot break it will be listened
> > to because I am known to be knowledgeable in how ciphers work...
>
> There's a name for this attitude. It's called the Aristotelean Fallacy
> -- the appeal to authority. It dominated science for centuries, and
> science suffered for it.

An appeal to authority is invalid under two conditions. First, if the
claim is subject to rigorous proof--making opinion irrelevant. Second,
if the authority appealed to is not a legitimate authority in a
relevant area. See
<http://www.nizkor.org/features/fallacies/appeal-to-authority.html.

When rigorous proof is not available, then the opinion of an expert
constitutes the best information to be had. Under that condition, the
best expert is the one with the longest experience and the most
successes.

> The fact that the best (only) standard we have for judging ciphers
> and their implementations is that of Brand Names indicates just how
> young/volatile/immature the field is.

Perhaps, but not necessarily. It is probable that Goedel's
Incompleteness Theorem implies that the strength of at least some
algorithms cannot be determined, even theoretically (forgive my
speculating aloud here). Further, it might turn out that all
'measurable' algorithms turn out to be weak--with some definition of
weak--implying that the non-measurable algorithms are the ONLY
interesting ones.

Remember, Fermat's last theorem went unproven for more than 350
years. Huge quantities of number-theoretic research arose directly out
of attempts to prove or disprove the theorem.

Remember, too, that many mathematical cranks turned up with "proofs"
of Fermat's theorem (and the four color theorem, and...). Call it
arrogant, but mathematicians tend to treat them with a priori

scepticism, given that 350 years of experts failed to turn up a
proof. One is quite justified in seriously doubting that Joe Blow from
Podunk has stumbled upon a solution.

Such considerations suggest, at least to me, that
"crypto-engineering", by which we might crank out ciphers of known
strength, is probably a pipe-dream.

BTW this example has a bearing on our confidence in RSA. It is doubted
that polynomial-time factoring of primes is possible, just as it is
doubted that NP = P. Further, it is conjectured that cracking RSA
without factoring is not possible (absent other data, such as
decryption timings). Why are these conjectures made? Because a
generation or so of experts and geniuses haven't resolved these
problems. If the NSA has, then they've almost certainly made one of
the great discoveries of the century. Of course, they're not talking.

```
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
Len Budney                  |  Designing a cipher takes only a
Maya Design Group           |  few minutes.  The only problem is
budney@maya.com             |  that almost all designs are junk.
                            |              -- Prof. Dan Bernstein
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
```

Subject: Re: Thought question: why do public ciphers use only simple ops like shift
and XOR?
Date: Thu, 22 Apr 1999 09:12:49 +0100
From: "Sam Simpson" <ssimpson@hertreg.ac.uk>
Message-ID: <371ed9e2.0@nnrp1.news.uk.psi.net>
References: <m3d80xwyh2.fsf@peregrine.maya.com>
Newsgroups: sci.crypt
Lines: 43

-----BEGIN PGP SIGNED MESSAGE-----
Hash: SHA1

Leonard R. Budney <budney@peregrine.maya.com> wrote in message
news:m3d80xwyh2.fsf@peregrine.maya.com...

<SNIP>

> BTW this example has a bearing on our confidence in RSA. It is
doubted
> that polynomial-time factoring of primes is possible, just as
it is
> doubted that NP = P. Further, it is conjectured that cracking
RSA
> without factoring is not possible (absent other data, such as
> decryption timings).

Actually, certain instances of RSA cannot be equivalent to the
underlying IFP (D.Boneh, R.Venkatesan, "Breaking RSA may not be
equivalent to factoring").

Cheers,

- --
Sam Simpson
Comms Analyst
http://www.scramdisk.clara.net/ for ScramDisk hard-drive
encryption & Delphi Crypto Components. PGP Keys available at the
same site.
If you're wondering why I don't reply to Sternlight, it's because
he's kill filed. See http://www.openpgp.net/FUD for why!

-----BEGIN PGP SIGNATURE-----
Version: 6.0.2ckt http://members.tripod.com/IRFaiad/

iQA/AwUBNx7Z/u0ty8FDP9tPEQJVjwCdElMbx8eOjPva0qOKAkCTzKte+MwAoMoE
PG95Mhvh0WP9lAZT5Sw5XwRC
=SIRn
-----END PGP SIGNATURE-----

Subject: Re: Thought question: why do public ciphers use only simple ops like shift
and XOR?
Date: 16 Apr 1999 17:21:22 -0400
From: juola@mathcs.duq.edu (Patrick Juola)
Message-ID: <7f89kigng1@quine.mathcs.duq.edu>
References: <37179b67.12809750@news.io.com>
Newsgroups: sci.crypt
Lines: 38

In article <37179b67.12809750@news.io.com>, Terry Ritter <ritter@io.com> wrote:
>
>On Fri, 16 Apr 1999 17:28:13 GMT, in
><37176a30.4219613@news.prosurfr.com>, in sci.crypt
>jsavard@tenMAPSONeerf.edmonton.ab.ca (John Savard) wrote:
>
>>[...]

>>- Also, since there are many insecure cipher designs floating around, one
>>can't just accept that a cipher is secure based on its designer's say-so.
>>Instead, what gives real confidence in a cipher design is that it has been
>>studied by experts who have failed to crack it, but who have come away from
>>their attempts with an understanding of the source of the design's
>>strengths.
>
>I dispute this. This is essentially what Schneier would have us
>believe, and it is false.
>
>The truth is that we *never* know the "real" strength of a cipher. No
>matter how much review or cryptanalysis a cipher gets, we only have
>the latest "upper bound" for strength. The lower bound is zero: Any
>cipher can fail at any time.
>
>Since we have only an upper bound for the strength of any cipher, any
>confidence we may have is no more than our own delusion. We wish and
>hope for cipher strength, and -- absent a specific proof otherwise --
>we gradually come to believe in it. But that does not make it true.

So you're suggesting that a cypher that has withstood years of
intensive analysis by professionals is *NO* better than a cypher
that has not been analyzed at all?

I don't believe this; in fact, I think it's total bullshit. It's
certainly true that you may not be able to *formalize* the difference
into a p-value, but you're committing a grievious error if you
think that something doesn't exist merely because you can't quantify
it.

-kitten

Subject: Re: Thought question: why do public ciphers use only simple ops like shift
and XOR?
Date: Fri, 16 Apr 1999 23:53:19 GMT
From: ritter@io.com (Terry Ritter)
Message-ID: <3717cd6d.25617381@news.io.com>
References: <7f89kigng1@quine.mathcs.duq.edu>
Newsgroups: sci.crypt
Lines: 53

On 16 Apr 1999 17:21:22 -0400, in <7f89kigng1@quine.mathcs.duq.edu>,
in sci.crypt juola@mathcs.duq.edu (Patrick Juola) wrote:

>[...]
>So you're suggesting that a cypher that has withstood years of
>intensive analysis by professionals is *NO* better than a cypher
>that has not been analyzed at all?

It is not provably better. And not provably better admits the
possibility of contradiction. So we do not know. Which means that
interpreting years of intensive analysis as strength is nothing more
than DELUSION. Cryptanalysis of any length whatsoever provides no
rational scientific indication of strength.

>I don't believe this;

It is not necessary for you to believe it: It is what it is.

>in fact, I think it's total bullshit.

Then you need to think about it more deeply.

>It's
>certainly true that you may not be able to *formalize* the difference
>into a p-value, but you're committing a grievious error if you
>think that something doesn't exist merely because you can't quantify
>it.

The issue is not the "formalization" of something we know but cannot
quantify, but rather something we actually do not know. When we
attempt to formalize what we really do not know we commit logical
error. In fact, I would say that this process is in some cases a
deliberate attempt to hide these issues from management, command staff
and the general user.

In some cases this process is a deliberate attempt to make
cryptanalysis seem more than it is, so that ciphers which have
"passed" (whatever that means) will be accepted as "strong," which
should never be done. We can see this in the path of the AES process,
which, presumably, gets us a "strong" cipher. We see NO attempt to
innovate constructions or protocols which give strength in the context
of ciphers which may be weak. Yet you would have us assume that
everyone knows that ciphers may be weak, and simply chooses to do
nothing about it.

Terry Ritter ritter@io.com http://www.io.com/~ritter/
Crypto Glossary http://www.io.com/~ritter/GLOSSARY.HTM

Subject: Re: Thought question: why do public ciphers use only simple ops like shift and XOR?
Date: Sat, 17 Apr 1999 02:28:52 GMT
From: fqkhuo@gmrvavvrcd.fl (ybizmt)
Message-ID: <slrn7hfsef.cc.fqkhuo@tpep.nofsozwovh.yq>
References: <3717cd6d.25617381@news.io.com>
Newsgroups: sci.crypt
Lines: 26

On Fri, 16 Apr 1999 23:53:19 GMT, Terry Ritter <ritter@io.com> wrote:
> It is not provably better. And not provably better admits the
> possibility of contradiction. So we do not know. Which means that
> interpreting years of intensive analysis as strength is nothing more
> than DELUSION. Cryptanalysis of any length whatsoever provides no
> rational scientific indication of strength.

Nor is it intended to. Who has ever claimed that analysis equals
strength in any field? It is intended to make you more confident
that something is strong. No one is saying it proves strength.
Not at least trying cryptanalysis on a cipher is stupid which
I'm sure you agree with.

> In some cases this process is a deliberate attempt to make
> cryptanalysis seem more than it is, so that ciphers which have
> "passed" (whatever that means) will be accepted as "strong," which
> should never be done. We can see this in the path of the AES process,
> which, presumably, gets us a "strong" cipher. We see NO attempt to
> innovate constructions or protocols which give strength in the context
> of ciphers which may be weak. Yet you would have us assume that
> everyone knows that ciphers may be weak, and simply chooses to do
> nothing about it.

Nice rant. Where are you going with this and how does it sell your
product?

Subject: Re: Thought question: why do public ciphers use only simple ops like shift and XOR?
Date: Sat, 17 Apr 1999 04:39:24 GMT
From: ritter@io.com (Terry Ritter)
Message-ID: <37181079.5255438@news.io.com>
References: <slrn7hfsef.cc.fqkhuo@tpep.nofsozwovh.yq>
Newsgroups: sci.crypt
Lines: 81

On Sat, 17 Apr 1999 02:28:52 GMT, in
<slrn7hfsef.cc.fqkhuo@tpep.nofsozwovh.yq>, in sci.crypt
fqkhuo@gmrvavvrcd.fl (ybizmt) wrote:

>On Fri, 16 Apr 1999 23:53:19 GMT, Terry Ritter <ritter@io.com> wrote:
>> It is not provably better. And not provably better admits the
>> possibility of contradiction. So we do not know. Which means that
>> interpreting years of intensive analysis as strength is nothing more
>> than DELUSION. Cryptanalysis of any length whatsoever provides no
>> rational scientific indication of strength.
>
>Nor is it intended to. Who has ever claimed that analysis equals
>strength in any field? It is intended to make you more confident
>that something is strong. No one is saying it proves strength.

Sure they are. As far as I know, Schneier's point has always been
that cryptanalysis is the way we know a cipher's strength. I'm sure
he would agree that this is not proof, but I do not agree that it says
anything at all. The implication that cryptanalysis would like to
promote is indeed that of tested strength.

>Not at least trying cryptanalysis on a cipher is stupid which
>I'm sure you agree with.

I do. But there is no one cryptanalysis. Indeed, there is no end to
it. But we do have to make an end before we can field anything. This
in itself tells us that cryptanalysis as certification is necessarily
incomplete.

Our main problem is that cryptanalysis does NOT say that there is no
simpler attack. It does NOT say that a well-examined cipher is secure
from your kid sister. Oh, many people will offer their opinion, but
you won't see many such a claims in scientific papers, because there
we expect actual facts, as opposed to wishes, hopes, and dreams.

Cryptanalysis does NOT give us an indication of how much effort our
Opponent will have to spend to break the cipher. Yet that is exactly
what the cryptanalytic process would like us to believe: That is why
we have the process of: 1) design a cipher, and 2) certify the
cipher by cryptanalysis. As I see it, the real opportunity for
cryptanalysis is as part of a dynamic and interactive cipher design
process, as opposed to final certification.

>> In some cases this process is a deliberate attempt to make
>> cryptanalysis seem more than it is, so that ciphers which have
>> "passed" (whatever that means) will be accepted as "strong," which
>> should never be done. We can see this in the path of the AES process,
>> which, presumably, gets us a "strong" cipher. We see NO attempt to
>> innovate constructions or protocols which give strength in the context
>> of ciphers which may be weak. Yet you would have us assume that
>> everyone knows that ciphers may be weak, and simply chooses to do
>> nothing about it.
>
>Nice rant.

Thanks. I suggest you learn it by heart if you intend to depend upon
cryptography.

>Where are you going with this and how does it sell your
>product?

This is my bit for public education.

I have no modern products. I do offer cryptographic consulting time,
and then I call it as I see it. I also own patented cryptographic
technology which could be useful in a wide range of ciphers.

I see no problem with someone promoting what they think is an advance
in the field, even if they will benefit. But when reasoning errors
are promoted which just happen to benefit one's business -- in fact, a
whole sub-industry -- some skepticism seems appropriate. Just once I
would like to see delusions promoted which produce *less* business.

Terry Ritter ritter@io.com http://www.io.com/~ritter/
Crypto Glossary http://www.io.com/~ritter/GLOSSARY.HTM

Subject: Re: Thought question: why do public ciphers use only simple ops like shift
and XOR?
Date: 18 Apr 99 02:05:37 GMT
From: jsavard@ecn.ab.ca ()
Message-ID: <37193df1.0@ecn.ab.ca>
References: <37181079.5255438@news.io.com>
Newsgroups: sci.crypt
Lines: 16

Terry Ritter (ritter@io.com) wrote:
: As I see it, the real opportunity for
: cryptanalysis is as part of a dynamic and interactive cipher design
: process, as opposed to final certification.

Two comments are warranted here.

- Since cryptanalysis represents the "hard" part of the work in designing
a cipher, this is why cipher designers should themselves know something
about cryptanalysis;

- And I think you can see why this design process actually _increases_ the
probability of a design which is strong against known attacks, but weak
against a future attack someone might discover.

John Savard

Subject: Re: Thought question: why do public ciphers use only simple ops like shift
and XOR?
Date: Sun, 18 Apr 1999 22:04:54 GMT
From: jsavard@tenMAPSONeerf.edmonton.ab.ca (John Savard)
Message-ID: <371a56a8.198396@news.prosurfr.com>
References: <37193df1.0@ecn.ab.ca>
Newsgroups: sci.crypt
Lines: 23

jsavard@ecn.ab.ca () wrote, in part:
>Terry Ritter (ritter@io.com) wrote:
>: As I see it, the real opportunity for
>: cryptanalysis is as part of a dynamic and interactive cipher design
>: process, as opposed to final certification.

>Two comments are warranted here.

>- Since cryptanalysis represents the "hard" part of the work in designing
>a cipher, this is why cipher designers should themselves know something
>about cryptanalysis;

>- And I think you can see why this design process actually _increases_ the
>probability of a design which is strong against known attacks, but weak
>against a future attack someone might discover.

I should note, though, that I basically agree with your point - and - but I simply
think that these two
arguments also need to be addressed.

John Savard (teenerf<-)

http://members.xoom.com/quadibloc/index.html

Subject: Re: Thought question: why do public ciphers use only simple ops like shift and XOR?
Date: Tue, 20 Apr 1999 22:03:33 GMT
From: ritter@io.com (Terry Ritter)
Message-ID: <371cf9af.7589747@news.io.com>
References: <37193df1.0@ecn.ab.ca>
Newsgroups: sci.crypt
Lines: 28

On 18 Apr 99 02:05:37 GMT, in <37193df1.0@ecn.ab.ca>, in sci.crypt
jsavard@ecn.ab.ca () wrote:

>Terry Ritter (ritter@io.com) wrote:
>: As I see it, the real opportunity for
>: cryptanalysis is as part of a dynamic and interactive cipher design
>: process, as opposed to final certification.
>
>Two comments are warranted here.
>
>- Since cryptanalysis represents the "hard" part of the work in designing
>a cipher, this is why cipher designers should themselves know something
>about cryptanalysis;

I agree.

>- And I think you can see why this design process actually _increases_ the
>probability of a design which is strong against known attacks, but weak
>against a future attack someone might discover.

You lost me on that one.

Terry Ritter ritter@io.com http://www.io.com/~ritter/
Crypto Glossary http://www.io.com/~ritter/GLOSSARY.HTM

Subject: Re: Thought question: why do public ciphers use only simple ops like shift and XOR?
Date: Wed, 21 Apr 1999 16:12:35 GMT
From: jsavard@tenMAPSONeerf.edmonton.ab.ca (John Savard)
Message-ID: <371df7b2.320404@news.prosurfr.com>
References: <371cf9af.7589747@news.io.com>
Newsgroups: sci.crypt
Lines: 27

ritter@io.com (Terry Ritter) wrote, in part:
>On 18 Apr 99 02:05:37 GMT, in <37193df1.0@ecn.ab.ca>, in sci.crypt
>jsavard@ecn.ab.ca () wrote:

>>- And I think you can see why this design process actually _increases_ the
>>probability of a design which is strong against known attacks, but weak
>>against a future attack someone might discover.

>You lost me on that one.

When testing a computer system, sometimes a small number of known bugs are deliberately introduced, so that, if not all of _those_ bugs are found, one has an indication that testing should continue (on the assumption that a similar proportion of the unknown bugs really being looked for have not been found yet either).

What I was thinking of here is that the cryptanalyst will find what he knows how to look for; and so, weaknesses beyond the reach of current cryptanalysis won't be found; but if a cipher designed by a non-cryptanalyst did not have a *single* known weakness (known to the cryptanalysts, not to the designer) then one might have grounds to hope (but, of course, not proof) that unknown weaknesses were scarce as well, while getting rid of the known weaknesses _specifically_ doesn't give any such hope.

John Savard (teneerf<-)
http://members.xoom.com/quadibloc/index.html

Subject: Re: Thought question: why do public ciphers use only simple ops like shift and XOR?
Date: Wed, 21 Apr 1999 18:59:11 GMT
From: ritter@io.com (Terry Ritter)
Message-ID: <371e1f94.6051889@news.io.com>
References: <371df7b2.320404@news.prosurfr.com>
Newsgroups: sci.crypt
Lines: 67

On Wed, 21 Apr 1999 16:12:35 GMT, in
<371df7b2.320404@news.prosurfr.com>, in sci.crypt
jsavard@tenMAPSONeerf.edmonton.ab.ca (John Savard) wrote:

>ritter@io.com (Terry Ritter) wrote, in part:
>>On 18 Apr 99 02:05:37 GMT, in <37193df1.0@ecn.ab.ca>, in sci.crypt
>>jsavard@ecn.ab.ca () wrote:
>
>>>- And I think you can see why this design process actually _increases_ the
>>>probability of a design which is strong against known attacks, but weak
>>>against a future attack someone might discover.
>
>>You lost me on that one.
>
>When testing a computer system, sometimes a small number of known bugs are
>deliberately introduced, so that, if not all of _those_ bugs are found, one
>has an indication that testing should continue (on the assumption that a
>similar proportion of the unknown bugs really being looked for have not
>been found yet either).

I believe this is generally called "error injection," and one problem with it is the assumption that the known errors are of the same nature as the unknown errors. Only then can we extrapolate from our results into the unknown. Basically what we measure is the effectiveness of the process which seeks that sort of error -- usually some sort of mechanical error like failing to use the result of some computation. This is not going to work very well when the errors are conceptual in the structure of the computation itself. Error injection is not very useful in asserting that we will get the correct answer to the original problem, and that is the unknown crypto area.

So this doesn't really help us.

>What I was thinking of here is that the cryptanalyst will find what he
>knows how to look for; and so, weaknesses beyond the reach of current
>cryptanalysis won't be found; but if a cipher designed by a
>non-cryptanalyst did not have a *single* known weakness (known to the
>cryptanalysts, not to the designer) then one might have grounds to hope
>(but, of course, not proof) that unknown weaknesses were scarce as well,
>while getting rid of the known weaknesses _specifically_ doesn't give any
>such hope.

The idea of a brand-new designer with a brand-new design in which no weakness can be found is a silly hope. I suppose it might happen, but it is not the way real things are designed and built. At the very best it is a wish, a dream, something disassociated with practical reality and the design of real things. And the failure of such exaggerated expectations often leads to a supposedly-justified demeaning of the designer as not meeting the goals of the field. This is essentially sick reasoning, because it sets up unreasonable goals, then reacts with staged regret when they are not met.

I claim the main use of cryptanalysis is in the give and take of a design process, not the end game of certification, which is what cryptanalysis cannot do. In fact, academic cryptanalysis generally only reports weakness -- few reports are published that no weakness was found. There is thus no basis even in open cryptography for knowing how many cryptanalytic attempts have been made unsuccessfully, or for taking advantage of the game when a new designer actually does have a design which has no known weakness.

Terry Ritter ritter@io.com http://www.io.com/~ritter/
Crypto Glossary http://www.io.com/~ritter/GLOSSARY.HTM

Subject: Re: Thought question: why do public ciphers use only simple ops like shift and XOR?
Date: 17 Apr 1999 16:32:27 -0400
From: juola@mathcs.duq.edu (Patrick Juola)
Message-ID: <7far4rhtf1@quine.mathcs.duq.edu>
References: <3717cd6d.25617381@news.io.com>
Newsgroups: sci.crypt
Lines: 48

In article <3717cd6d.25617381@news.io.com>, Terry Ritter <ritter@io.com> wrote:
>
>On 16 Apr 1999 17:21:22 -0400, in <7f89kigng1@quine.mathcs.duq.edu>,
>in sci.crypt juola@mathcs.duq.edu (Patrick Juola) wrote:
>
>>[...]
>>So you're suggesting that a cypher that has withstood years of
>>intensive analysis by professionals is *NO* better than a cypher
>>that has not been analyzed at all?
>
>It is not provably better. And not provably better admits the
>possibility of contradiction.

But not-provable is not the same as unknown.

I don't know that Pittsburgh won't be hit by a devastating hurricane in the next month.

But I've got a bright crisp $20 in my pocket that says that it won't.

In a philosophical sense, "knowledge" is a "justified true belief"; I don't have *proof* that Pittsburgh won't be hit by a hurricane,

but I can produce lots and lots of justification.

> So we do not know. Which means that
>interpreting years of intensive analysis as strength is nothing more
>than DELUSION. Cryptanalysis of any length whatsoever provides no
>rational scientific indication of strength.

Interesting. So your "rational scientific indication" is that we've
got no way of figuring out which side of my Pittsburgh weather bet
is the smart one?

>>I don't believe this;
>
>It is not necessary for you to believe it: It is what it is.
>
>
>>in fact, I think it's total bullshit.
>
>Then you need to think about it more deeply.

I just did. It's still total bullshit.

Knowledge doesn't require proof. Belief doesn't require knowledge.
Confidence doesn't even require belief.

        -kitten

Subject: Re: Thought question: why do public ciphers use only simple ops like shift
and XOR?
Date: Sat, 17 Apr 1999 23:40:04 GMT
From: ritter@io.com (Terry Ritter)
Message-ID: <37191bc9.2524456@news.io.com>
References: <7far4rhtf1@quine.mathcs.duq.edu>
Newsgroups: sci.crypt
Lines: 85

On 17 Apr 1999 16:32:27 -0400, in <7far4rhtf1@quine.mathcs.duq.edu>,
in sci.crypt juola@mathcs.duq.edu (Patrick Juola) wrote:

>In article <3717cd6d.25617381@news.io.com>, Terry Ritter <ritter@io.com>
wrote:
>>
>>On 16 Apr 1999 17:21:22 -0400, in <7f89kigng1@quine.mathcs.duq.edu>,
>>in sci.crypt juola@mathcs.duq.edu (Patrick Juola) wrote:
>>
>>>[...]
>>>So you're suggesting that a cypher that has withstood years of
>>>intensive analysis by professionals is *NO* better than a cypher
>>>that has not been analyzed at all?
>>
>>It is not provably better. And not provably better admits the
>>possibility of contradiction.
>
>But not-provable is not the same as unknown.
>
>I don't know that Pittsburgh won't be hit by a devastating hurricane
>in the next month.
>
>But I've got a bright crisp $20 in my pocket that says that it won't.

Which means to me that you have some understanding of the risk of
hurricanes in Pittsburgh. You get this understanding from reported
reality.

Unfortunately, neither you nor anyone else can have a similar
understanding of the risk of cipher failure -- there is no reporting
of cipher failure. There is instead every effort made to keep that
information secret, and in fact to generate false reporting to buoy
your unfounded delusion of strength.

>In a philosophical sense, "knowledge" is a "justified true belief";
>I don't have *proof* that Pittsburgh won't be hit by a hurricane,
>but I can produce lots and lots of justification.

Too bad we cannot do the same for a cipher.

>> So we do not know. Which means that
>>interpreting years of intensive analysis as strength is nothing more
>>than DELUSION. Cryptanalysis of any length whatsoever provides no
>>rational scientific indication of strength.
>
>Interesting. So your "rational scientific indication" is that we've
>got no way of figuring out which side of my Pittsburgh weather bet
>is the smart one?

Nonsense. Knowing the past weather in Pittsbugh is possible: Knowing
the past strength of a cipher is not.

>>>I don't believe this;

>>
>>It is not necessary for you to believe it: It is what it is.
>>
>>
>>>in fact, I think it's total bullshit.
>>
>>Then you need to think about it more deeply.
>
>I just did. It's still total bullshit.

Then you need to think about it even more deeply.

>Knowledge doesn't require proof. Belief doesn't require knowledge.
>Confidence doesn't even require belief.

Fine. I will grant that you can be confident completely independent of reality. Oddly, I assumed that we were talking Science here.

RATIONAL confidence requires a quantification of risk, even if only as a handwave generality. But that is not available in ciphers. Until we have a complete theory of strength, or a complete theory of cryptanalysis, we have no basis by which to judge the risk we take by using any particular cipher.

Terry Ritter ritter@io.com http://www.io.com/~ritter/
Crypto Glossary http://www.io.com/~ritter/GLOSSARY.HTM

Subject: Re: Thought question: why do public ciphers use only simple ops like shift and XOR?
Date: 18 Apr 99 01:55:36 GMT
From: jsavard@ecn.ab.ca ()
Message-ID: <37193b98.0@ecn.ab.ca>
References: <3717cd6d.25617381@news.io.com>
Newsgroups: sci.crypt
Lines: 31

Terry Ritter (ritter@io.com) wrote:

: On 16 Apr 1999 17:21:22 -0400, in <7f89kigng1@quine.mathcs.duq.edu>,
: in sci.crypt juola@mathcs.duq.edu (Patrick Juola) wrote:

: >[...]
: >So you're suggesting that a cypher that has withstood years of
: >intensive analysis by professionals is *NO* better than a cypher
: >that has not been analyzed at all?

: It is not provably better. And not provably better admits the
: possibility of contradiction. So we do not know. Which means that
: interpreting years of intensive analysis as strength is nothing more
: than DELUSION. Cryptanalysis of any length whatsoever provides no
: rational scientific indication of strength.

Yes and no.

Your point is valid, however, what do we do if there is no way to obtain a lower bound on the strength of a cipher? I fear this is quite possible: proving a cipher is strong against attacks we can't even imagine seems to me to be equivalent to solving the halting problem.

Then it does make sense to look at the upper bound, because it's one of the few indications we have. But it also makes sense - and here, I think, we come closer to agreement - not to put too much faith in that upper bound, and to add constructs of different types, and constructs that seem like any mathematical tools to analyze them which would be useful for cryptanalysts are *far* in advance of the state of current knowledge.

John Savard

Subject: Re: Thought question: why do public ciphers use only simple ops like shift and XOR?
Date: Tue, 20 Apr 1999 22:03:47 GMT
From: ritter@io.com (Terry Ritter)
Message-ID: <371cf9b7.7597561@news.io.com>
References: <37193b98.0@ecn.ab.ca>
Newsgroups: sci.crypt
Lines: 55

On 18 Apr 99 01:55:36 GMT, in <37193b98.0@ecn.ab.ca>, in sci.crypt
jsavard@ecn.ab.ca () wrote:

>Terry Ritter (ritter@io.com) wrote:
>
>: On 16 Apr 1999 17:21:22 -0400, in <7f89kigng1@quine.mathcs.duq.edu>,
>: in sci.crypt juola@mathcs.duq.edu (Patrick Juola) wrote:
>
>: >[...]
>: >So you're suggesting that a cypher that has withstood years of
>: >intensive analysis by professionals is *NO* better than a cypher
>: >that has not been analyzed at all?

>
>: It is not provably better. And not provably better admits the
>: possibility of contradiction. So we do not know. Which means that
>: interpreting years of intensive analysis as strength is nothing more
>: than DELUSION. Cryptanalysis of any length whatsoever provides no
>: rational scientific indication of strength.
>
>Yes and no.
>
>Your point is valid, however, what do we do if there is no way to obtain a
>lower bound on the strength of a cipher? I fear this is quite possible:

I agree.

>proving a cipher is strong against attacks we can't even imagine seems to
>me to be equivalent to solving the halting problem.

We have the testimony of 50 years of mathematical cryptography which
has not achieved the Holy Grail. I just think reality is trying to
tell us something.

>Then it does make sense to look at the upper bound, because it's one of
>the few indications we have.

No. Completely false. I see no reason why the upper bound should
have any correlation at all to the lower bound.

In any security audit, we have to consider the worst case attacks, not
just the ones we expect, and not just the ones we tried.

>But it also makes sense - and here, I think,
>we come closer to agreement - not to put too much faith in that upper
>bound, and to add constructs of different types, and constructs that seem
>like any mathematical tools to analyze them which would be useful for
>cryptanalysts are *far* in advance of the state of current knowledge.

I'm not sure I understand this fully.

Terry Ritter ritter@io.com http://www.io.com/~ritter/
Crypto Glossary http://www.io.com/~ritter/GLOSSARY.HTM

Subject: Re: Thought question: why do public ciphers use only simple ops like shift
and XOR?
Date: Wed, 21 Apr 1999 16:21:01 GMT
From: jsavard@tenMAPSONeerf.edmonton.ab.ca (John Savard)
Message-ID: <371df919.679323@news.prosurfr.com>
References: <371cf9b7.7597561@news.io.com>
Newsgroups: sci.crypt
Lines: 37

ritter@io.com (Terry Ritter) wrote, in part:
>On 18 Apr 99 01:55:36 GMT, in <37193b98.0@ecn.ab.ca>, in sci.crypt
>jsavard@ecn.ab.ca () wrote:

>>Then it does make sense to look at the upper bound, because it's one of
>>the few indications we have.

>No. Completely false. I see no reason why the upper bound should
>have any correlation at all to the lower bound.

It will definitely be higher than the lower bound, but yes, it doesn't
prevent the lower bound from being low.

>In any security audit, we have to consider the worst case attacks, not
>just the ones we expect, and not just the ones we tried.

Any security audit will have to include a disclaimer that the true security
of the cipher systems used is essentially unknowable, but even real-world
financial audits do routinely include various sorts of disclaimer.

>>But it also makes sense - and here, I think,
>>we come closer to agreement - not to put too much faith in that upper
>>bound, and to add constructs of different types, and constructs that seem
>>like any mathematical tools to analyze them which would be useful for
>>cryptanalysts are *far* in advance of the state of current knowledge.

>I'm not sure I understand this fully.

Given that a cipher highly resistant to known attacks (i.e., differential
cryptanalysis) _could_ still be very weak, as far as we know, what can we
do about it? The closest thing to a sensible suggestion I can make is this:
make our ciphers stronger (that is, use more rounds) and more intrinsically
difficult to analyze (use complicated, highly nonlinear, constructs) than
the known attacks indicate is necessary.

John Savard (teneerf<-)
http://members.xoom.com/quadibloc/index.html

Subject: Re: Thought question: why do public ciphers use only simple ops like shift
and XOR?
Date: Wed, 21 Apr 1999 18:59:23 GMT

From: ritter@io.com (Terry Ritter)
Message-ID: <371e2003.6163199@news.io.com>
References: <371df919.679323@news.prosurfr.com>
Newsgroups: sci.crypt
Lines: 73

On Wed, 21 Apr 1999 16:21:01 GMT, in
<371df919.679323@news.prosurfr.com>, in sci.crypt
jsavard@tenMAPSONeerf.edmonton.ab.ca (John Savard) wrote:

>ritter@io.com (Terry Ritter) wrote, in part:
>>On 18 Apr 99 01:55:36 GMT, in <37193b98.0@ecn.ab.ca>, in sci.crypt
>>jsavard@ecn.ab.ca () wrote:
>
>>>Then it does make sense to look at the upper bound, because it's one of
>>>the few indications we have.
>
>>No. Completely false. I see no reason why the upper bound should
>>have any correlation at all to the lower bound.
>
>It will definitely be higher than the lower bound, but yes, it doesn't
>prevent the lower bound from being low.
>
>>In any security audit, we have to consider the worst case attacks, not
>>just the ones we expect, and not just the ones we tried.
>
>Any security audit will have to include a disclaimer that the true security
>of the cipher systems used is essentially unknowable, but even real-world
>financial audits do routinely include various sorts of disclaimer.

I think you will find that financial disclaimers are not to avoid
responsibility for the financial service supplied. For example, an
audit disclaimer might say that the audit results were correct,
provided the supplied accounting information was correct. But that
is something which is, at least in principle, verifiable.

We don't have financial disclaimers which say that the audit is 90
percent certain to be correct, which is the sort of thing you might
like to think that cryptanalytic certification could at least do,
since it cannot provide certainty. But the very idea makes no sense.
The very companies that need the best auditing might also be the most
deceptive and able to hide their manipulations. There is no useful
"average" company, and so no useful statistics. Every case is
different.

>>>But it also makes sense - and here, I think,
>>>we come closer to agreement - not to put too much faith in that upper
>>>bound, and to add constructs of different types, and constructs that seem
>>>like any mathematical tools to analyze them which would be useful for
>>>cryptanalysts are *far* in advance of the state of current knowledge.
>
>>I'm not sure I understand this fully.
>
>Given that a cipher highly resistant to known attacks (i.e., differential
>cryptanalysis) _could_ still be very weak, as far as we know, what can we
>do about it? The closest thing to a sensible suggestion I can make is this:
>make our ciphers stronger (that is, use more rounds) and more intrinsically
>difficult to analyze (use complicated, highly nonlinear, constructs) than
>the known attacks indicate is necessary.

We could hardly disagree more.

I find "rounds" (the repeated application of the same operation) silly
and I don't use them. I do use "layers" in which different operations
are applied in each layer.

And I think that making a cipher more difficult to analyze can only
benefit the Opponents who have more resources for analysis.
Personally, I try to make ciphers as conceptually *simple* as possible
(though not simpler). Simple does not mean weak; simple means
appropriately decomposing the cipher into relatively few types of
substantial subcomponent which can be understood on their own, then
using those components in clear, structured ways.

Terry Ritter ritter@io.com http://www.io.com/~ritter/
Crypto Glossary http://www.io.com/~ritter/GLOSSARY.HTM

Subject: Re: Thought question: why do public ciphers use only simple ops like shift
and XOR?
Date: Wed, 21 Apr 1999 23:41:13 GMT
From: jsavard@tenMAPSONeerf.edmonton.ab.ca (John Savard)
Message-ID: <371e59c7.25432288@news.prosurfr.com>
References: <371e2003.6163199@news.io.com>
Newsgroups: sci.crypt
Lines: 53

ritter@io.com (Terry Ritter) wrote, in part:
>jsavard@tenMAPSONeerf.edmonton.ab.ca (John Savard) wrote:

>>Given that a cipher highly resistant to known attacks (i.e., differential
>>cryptanalysis) _could_ still be very weak, as far as we know, what can we

>>do about it? The closest thing to a sensible suggestion I can make is this:
>>make our ciphers stronger (that is, use more rounds) and more intrinsically
>>difficult to analyze (use complicated, highly nonlinear, constructs) than
>>the known attacks indicate is necessary.

>We could hardly disagree more.

>I find "rounds" (the repeated application of the same operation) silly
>and I don't use them. I do use "layers" in which different operations
>are applied in each layer.

>And I think that making a cipher more difficult to analyze can only
>benefit the Opponents who have more resources for analysis.
>Personally, I try to make ciphers as conceptually *simple* as possible
>(though not simpler). Simple does not mean weak; simple means
>appropriately decomposing the cipher into relatively few types of
>substantial subcomponent which can be understood on their own, then
>using those components in clear, structured ways.

It certainly does make sense to understand the parts of a cipher, to ensure
that the cipher is providing, as a minimum, some basic level of "security":
that is, for example, one might know that one's cipher is at least as
secure as DES, even if one doesn't know for sure that the effort required
to break DES is not trivial.

The original poster - Sundial Services - praised your Dynamic Substitution
because it "buries a lot more information" than ordinary designs, and this
is the sort of thing I'm thinking of. When I got past his first paragraph,
where he seemed to have forgotten about S-boxes, and saw that DynSub and
the SIGABA were the kinds of designs he praised, I saw that the kinds of
ciphers that appeal to him were the same ones as appeal intuitively to me.

Precisely because you have noted that we don't have a way to put a good
lower bound on the effort required to break a cipher, I find it hard to
think that I could achieve the goal, for a cipher, that is indeed
appropriate for a scientific theory, of making it "as simple as possible,
but no simpler"; if I am totally in the dark about how strong a cipher
really is, and how astute my adversaries are, that seems an inadvisable
goal, because I can never know what is necessary.

Since I have an upper bound instead of a lower bound, unless there is some
way to resolve that problem, and your researches may well achieve something
relevant, even if not a total solution, all I can do is try for a generous
margin of safety. True, it's not proof. But proof isn't available, except
for the one-time pad.

John Savard (teneerf<-)
http://members.xoom.com/quadibloc/index.html

Subject: Re: Thought question: why do public ciphers use only simple ops like shift
and XOR?
Date: Sat, 17 Apr 1999 00:38:15 -0600
From: jgfunj@vgrknf.arg (wtshaw)
Message-ID: <jgfunj-1704990038160001@dial-243-079.itexas.net>
References: <37179b67.12809750@news.io.com>
Newsgroups: sci.crypt
Lines: 28

In article <37179b67.12809750@news.io.com>, ritter@io.com (Terry Ritter) wrote:
>
> On the other hand, I have been pioneering the use of scalable
> technology which, presumably, can be scaled down to a level which can
> be investigated experimentally. The last I heard, experimentation was
> still considered a rational basis for the understanding of reality.
> Indeed, one might argue that in the absence of theoretical strength
> for *any* cipher, experimentation is about all we have. But note how
> little of it we see.
>
It's at least good science, beyond making lots of sense.
.....
>
> And in this way we can have hundreds or thousands of different
> ciphers, with more on the way all the time.

I resemble that remark. Better dust off the ole compiler again. More
dumb ciphers on the way...

> .....The result is that our Opponents must
> invest far more to get far less, and this advantage does not depend
> upon the delusion of strength which is all that cryptanalysis can
> provide.
>
It's aways difficult to stop a wave, be it composed of hoards of
combatants or algorithms.
--
Too much of a good thing can be much worse than none.

Subject: Re: Thought question: why do public ciphers use only simple ops like shift
and XOR?
Date: Mon, 19 Apr 1999 20:15:32 GMT
From: jsavard@tenMAPSONeerf.edmonton.ab.ca (John Savard)
Message-ID: <371b8ba8.16131590@news.prosurfr.com>

References: <37179b67.12809750@news.io.com>
Newsgroups: sci.crypt
Lines: 116

ritter@io.com (Terry Ritter) wrote, in part:
>jsavard@tenMAPSONeerf.edmonton.ab.ca (John Savard) wrote:

>>- Also, since there are many insecure cipher designs floating around, one
>>can't just accept that a cipher is secure based on its designer's say-so.
>>Instead, what gives real confidence in a cipher design is that it has been
>>studied by experts who have failed to crack it, but who have come away from
>>their attempts with an understanding of the source of the design's
>>strengths.

>I dispute this. This is essentially what Schneier would have us
>believe, and it is false.

>The truth is that we *never* know the "real" strength of a cipher. No
>matter how much review or cryptanalysis a cipher gets, we only have
>the latest "upper bound" for strength. The lower bound is zero: Any
>cipher can fail at any time.

I agree with you that we don't have a way to prove that a cipher really is
strong. But cryptanalysis still gives the best confidence currently
available.

>It is not, frankly, the role of the innovator to educate the
>academics, or even to serve technology to them on a silver platter.
>In the end, academic reputation comes from reality, and the reality is
>that many crypto academics avoid anything new which does not have an
>academic source. The consequence is that they simply do not have the
>background to judge really new designs.

That is true: the desires of the academic community aren't a valid excuse
for compromising one's cipher designs.

>Upon encountering a new design, anyone may choose to simplify that
>design and then report results from that simplification. This is done
>all the time. It is not necessary for an innovator to make a
>simplified design for this purpose.

And that is one of the reasons why.

>On the other hand, I have been pioneering the use of scalable
>technology which, presumably, can be scaled down to a level which can
>be investigated experimentally. The last I heard, experimentation was
>still considered a rational basis for the understanding of reality.
>Indeed, one might argue that in the absence of theoretical strength
>for *any* cipher, experimentation is about all we have. But note how
>little of it we see.

Are you drawing a distinction between "experimental investigation" and
"cryptanalysis"? If so, it would appear you are saying that there is an
additional method for obtaining some additional, though still imperfect,
confidence in a cipher design.

>>Plus, the risk that one's adversary is a hacker of the future with a very
>>powerful desktop computer seems much greater than the risk that one's
>>adversary will be an accomplished cryptanalyst, able to exploit the most
>>subtle flaws in an over-elaborate design.

>But we don't know our Opponents! If we have to estimate their
>capabilities, I think we are necessarily forced into assuming that
>they are more experienced, better equipped, have more time, are better
>motivated, and -- yes -- are even smarter than we are. There is
>ample opportunity for them to exploit attacks of which we have no
>inkling at all.

Most cipher users are more worried about their communications being read by
the typical computer hacker than by the NSA.

I suppose it's possible that one day a giant EFT heist will be pulled off
by retired NSA personnel, but that's the sort of thing which happens far
more often as the plot for a movie than in real life.

The problem is, of course, that if one has data that should remain secret
for 100 years, one does have to face advances in cryptanalytic
knowledge...as well as _unimaginable_ advances in computer power.

>>I believe it to be possible and useful to develop a design methodology -
>>mainly involving the cutting and pasting of pieces from proven cipher
>>designs - to enable a reasonably qualified person who, however, falls short
>>of being a full-fleged cryptographer, to design his own block cipher, and
>>thereby obtain additional and significant benefits in resistance to
>>cryptanalytic attack by having an unknown and unique algorithm.

>And in this way we can have hundreds or thousands of different
>ciphers, with more on the way all the time. That means that we can
>divide the worth of our information into many different ciphers, so
>that if any one fails, only a fraction of messages are exposed. It
>also means that *any* Opponent must keep up with new ciphers and
>analyze and possibly break each, then design a program, or build new
>hardware to exploit it. We can make good new ciphers cheaper than
>they can possibly be broken. The result is that our Opponents must
>invest far more to get far less, and this advantage does not depend

>upon the delusion of strength which is all that cryptanalysis can
>provide.

>>I don't deny that there are pitfalls looming in such an approach; if
>>something is left out of the methodology, or if it isn't conscientiously
>>used, people could easily wind up using weak designs and having a false
>>sense of security. I just think the problems can be addressed, and the
>>potential benefits are worth the attempt.

>Neat.

And of course, I must confess that my present efforts in this direction have not gotten to the point of providing an explicit "toolkit". I've contented myself with explaining, in my web site, a large number of historical designs - with a very limited discussion of cryptanalysis - and I've illustrated how an amateur might design a cipher only by example, with the ciphers of my Quadibloc series, as well as various ideas in the conclusions sections of the first four chapters.

Right now, although my web site is educational, it's also fairly light and entertaining as well: I haven't tried to trouble the reader with any difficult math, for example.

John Savard (teenerf<-)
http://members.xoom.com/quadibloc/index.html

Subject: Re: Thought question: why do public ciphers use only simple ops like shift and XOR?
Date: Tue, 20 Apr 1999 04:24:33 GMT
From: ritter@io.com (Terry Ritter)
Message-ID: <371c014c.3018295@news.io.com>
References: <371b8ba8.16131590@news.prosurfr.com>
Newsgroups: sci.crypt
Lines: 107

On Mon, 19 Apr 1999 20:15:32 GMT, in
<371b8ba8.16131590@news.prosurfr.com>, in sci.crypt
jsavard@tenMAPSONeerf.edmonton.ab.ca (John Savard) wrote:

>ritter@io.com (Terry Ritter) wrote, in part:
>[...]
>>The truth is that we *never* know the "real" strength of a cipher. No
>>matter how much review or cryptanalysis a cipher gets, we only have
>>the latest "upper bound" for strength. The lower bound is zero: Any
>>cipher can fail at any time.
>
>I agree with you that we don't have a way to prove that a cipher really is
>strong. But cryptanalysis still gives the best confidence currently
>available.

I guess I dispute "confidence." Confidence and Trust and Reliability are exactly what we do not have. I cannot say it more clearly: cryptanalysis gives us no lower bound to strength.

As an engineer growing up with an engineer dad, I have lived with bounded specifications most of my life. These bounds are what we pay for in products; this is the performance the manufacturer guarantees. I suppose like me most buyers have been caught at least once by the consequences getting the cheapest part on the basis of "typical" specs instead of "worst case." But "typical" is all cryptanalysis tells us. Depending on that will sink us, sooner or later.

>[...]
>>On the other hand, I have been pioneering the use of scalable
>>technology which, presumably, can be scaled down to a level which can
>>be investigated experimentally. The last I heard, experimentation was
>>still considered a rational basis for the understanding of reality.
>>Indeed, one might argue that in the absence of theoretical strength
>>for *any* cipher, experimentation is about all we have. But note how
>>little of it we see.
>
>Are you drawing a distinction between "experimental investigation" and
>"cryptanalysis"? If so, it would appear you are saying that there is an
>additional method for obtaining some additional, though still imperfect,
>confidence in a cipher design.

We were OK up to the "c" word: I assert that we *can* have no confidence in a cipher. We have no way to prove strength. Any strength we assume is based upon the conceit that all others are just as limited in their capabilities as we are. Drawing conclusions by wishing and hoping the other guy is at least as dumb as us is not my idea of good cryptography.

I do make a distinction (which probably should not exist) between "theoretical" or "equation-based" or "academic" cryptography and experimental investigation. I suppose this is really much like the difference between math and applied math, with much of the same theoretically friendly antagonism.

It is clear that we may never have a provable theory of strength. This may mean that our only possible avenue toward certainty is some

sort of exhaustive test. Surely we cannot imagine such testing of a full-size cipher. But if we can scale that same design down, in the same way that small integers work like large ones, maybe we can work with large enough samples of the full population to be able to draw reasonable experimental conclusions.

>>>Plus, the risk that one's adversary is a hacker of the future with a very
>>>powerful desktop computer seems much greater than the risk that one's
>>>adversary will be an accomplished cryptanalyst, able to exploit the most
>>>subtle flaws in an over-elaborate design.
>
>>But we don't know our Opponents! If we have to estimate their
>>capabilities, I think we are necessarily forced into assuming that
>>they are more experienced, better equipped, have more time, are better
>>motivated, and -- yes -- are even smarter than we are. There is
>>ample opportunity for them to exploit attacks of which we have no
>>inkling at all.
>
>Most cipher users are more worried about their communications being read by
>the typical computer hacker than by the NSA.
>
>I suppose it's possible that one day a giant EFT heist will be pulled off
>by retired NSA personnel, but that's the sort of thing which happens far
>more often as the plot for a movie than in real life.
>
>The problem is, of course, that if one has data that should remain secret
>for 100 years, one does have to face advances in cryptanalytic
>knowledge...as well as _unimaginable_ advances in computer power.

I wrote in a post which I did not send that if *only* NSA could read my mail, the way it is now, I would not much care. Of course things change in politics, and my view could change as well. But for me, NSA is really just an illustration of the abstract threat.

As I understand security, one of the worst things we can do is to make assumptions about our Opponents which do not represent their full threat capabilities. ("Never underestimate your opponent.") Because of this I am not interested in identifying a cipher Opponent, unless in the process I can identify them as the absolute worst threat and know their capabilities as well. This is obviously impossible. So if we are to enforce our security despite the actions and intents of others, we must assume our Opponents are far more powerful than we know, then learn to deal with that threat.

Terry Ritter ritter@io.com http://www.io.com/~ritter/
Crypto Glossary http://www.io.com/~ritter/GLOSSARY.HTM

Subject: Re: Thought question: why do public ciphers use only simple ops like shift and XOR?
Date: Tue, 20 Apr 1999 19:20:24 +0200
From: Mok-Kong Shen <mok-kong.shen@stud.uni-muenchen.de>
Message-ID: <371CB758.E30A081B@stud.uni-muenchen.de>
References: <371c014c.3018295@news.io.com>
Newsgroups: sci.crypt
Lines: 17

Terry Ritter wrote:
>

> I guess I dispute "confidence." Confidence and Trust and Reliability
> are exactly what we do not have. I cannot say it more clearly:
> cryptanalysis gives us no lower bound to strength.

No intention to take part in the current discussion. But the word 'lower bound' raised association in my mind to an interesting sentence that A. Salomaa wrote (1990):

    There are no provable lower bounds for the amount of work
    of a cryptanalyst analyzing a public-key cryptosystem.

M. K. Shen
http://www.stud.uni-muenchen.de/~mok-kong.shen/ (Updated: 12 Apr 99)

Subject: Re: Thought question: why do public ciphers use only simple ops like shift and XOR?
Date: 23 Apr 1999 05:39:45 GMT
From: olson@umbc.edu (Bryan G. Olson; CMSC (G))
Message-ID: <7fp131$dg1$1@news.umbc.edu>
References: <37179b67.12809750@news.io.com>
Newsgroups: sci.crypt
Lines: 51

Terry Ritter (ritter@io.com) wrote:

[...]
: It may be unfortunate for academic cryptographers that a wide variety
: of new techniques are pioneered by non-academics. But those
: techniques exist nevertheless, and to the extent that academics do not

: investigate them, those academics are not up with the state of the
: art.

: It is not, frankly, the role of the innovator to educate the
: academics, or even to serve technology to them on a silver platter.
: In the end, academic reputation comes from reality, and the reality is
: that many crypto academics avoid anything new which does not have an
: academic source.

This impression of the academic crypto community as a closed
club that ignores the work of outsiders is flat out false.
Consider power and timing analysis - the entire area came
from the crypto left-field and was pioneered by a recent grad
with a B.A. in biology. The work was good, so now he's one
of those respected cryptologists. The various attacks I've
heard on academics are invariably by those whose work is
simply not of the same caliber.

For an example of an idea the crypto community has ignored
because it is truly dreadful:

[...]
: And in this way we can have hundreds or thousands of different
: ciphers, with more on the way all the time. That means that we can
: divide the worth of our information into many different ciphers, so
: that if any one fails, only a fraction of messages are exposed.

Absurdly naive. In any real project or real enterprise, the
same information is carried by many, many messages. The degree
of protection of any piece of intelligence is that of the
weakest of the systems carrying it.

: It
: also means that *any* Opponent must keep up with new ciphers and
: analyze and possibly break each, then design a program, or build new
: hardware to exploit it. We can make good new ciphers cheaper than
: they can possibly be broken. The result is that our Opponents must
: invest far more to get far less, and this advantage does not depend
: upon the delusion of strength which is all that cryptanalysis can
: provide.

Nonsense. The attacker just waits for the information he wants
to be transmitted under a system he can break.

--Bryan

Subject: Re: Thought question: why do public ciphers use only simple ops like shift
and XOR?
Date: Fri, 23 Apr 1999 21:23:23 GMT
From: jsavard@tenMAPSONeerf.edmonton.ab.ca (John Savard)
Message-ID: <3720e200.23217001@news.prosurfr.com>
References: <7fp131$dg1$1@news.umbc.edu>
Newsgroups: sci.crypt
Lines: 51

olson@umbc.edu (Bryan G. Olson; CMSC (G)) wrote, in part:

>This impression of the academic crypto community as a closed
>club that ignores the work of outsiders is flat out false.
>Consider power and timing analysis - the entire area came
>from the crypto left-field and was pioneered by a recent grad
>with a B.A. in biology. The work was good, so now he's one
>of those respected cryptologists. The various attacks I've
>heard on academics are invariably by those whose work is
>simply not of the same caliber.

I have every respect for the advanced work done by people such as Eli Biham
or David Wagner. And you're absolutely right that cryptography, like many
other fields, has its cranks and quacks.

However, I don't think it's appropriate to automatically conclude that
everyone who expresses concern about the way in which the public
cryptography field is going is necessarily a crank. For example, if even a
layperson looks at DES, or IDEA, or SERPENT, and expresses the opinion that
these designs all seem too regular, too repetitious, so that some form of
analysis at least seems like it may be someday possible - well, if that is
such a silly notion, what are you going to say to the people who designed
MARS, who happen to be the among the well-qualified?

>For an example of an idea the crypto community has ignored
>because it is truly dreadful:

>[...]
>: And in this way we can have hundreds or thousands of different
>: ciphers, with more on the way all the time. That means that we can
>: divide the worth of our information into many different ciphers, so
>: that if any one fails, only a fraction of messages are exposed.

>Absurdly naive. In any real project or real enterprise, the
>same information is carried by many, many messages. The degree
>of protection of any piece of intelligence is that of the
>weakest of the systems carrying it.

While that is true, that just means that, for internal encryption in an company data with ciphers their employer does not trust.

For a program of the PGP type, that lets people exchange E-Mail with other private individuals, allowing each party to specify a choice of preferred ciphers, and yet interoperate within the framework of using the same program, this sort of thing is a good idea.

'Dreadful' is not the same as 'not everywhere applicable'.

John Savard (teneerf<-)
http://members.xoom.com/quadibloc/index.html

Subject: Re: Thought question: why do public ciphers use only simple ops like shift and XOR?
Date: 25 Apr 1999 10:58:07 GMT
From: olson@umbc.edu (Bryan G. Olson; CMSC (G))
Message-ID: <7fusfv$as8$1@news.umbc.edu>
References: <3720e200.23217001@news.prosurfr.com>
Newsgroups: sci.crypt
Lines: 79

John Savard (jsavard@tenMAPSONeerf.edmonton.ab.ca) wrote:
: olson@umbc.edu (Bryan G. Olson; CMSC (G)) wrote, in part:

: >This impression of the academic crypto community as a closed
: >club that ignores the work of outsiders is flat out false.
: >Consider power and timing analysis - the entire area came
: >from the crypto left-field and was pioneered by a recent grad
: >with a B.A. in biology. The work was good, so now he's one
: >of those respected cryptologists. The various attacks I've
: >heard on academics are invariably by those whose work is
: >simply not of the same caliber.

: I have every respect for the advanced work done by people such as Eli Biham
: or David Wagner. And you're absolutely right that cryptography, like many
: other fields, has its cranks and quacks.

: However, I don't think it's appropriate to automatically conclude that
: everyone who expresses concern about the way in which the public
: cryptography field is going is necessarily a crank. For example, if even a
: layperson looks at DES, or IDEA, or SERPENT, and expresses the opinion that
: these designs all seem too regular, too repetitious, so that some form of
: analysis at least seems like it may be someday possible - well, if that is
: such a silly notion, what are you going to say to the people who designed
: MARS, who happen to be the among the well-qualified?

Quite right, but as I understood Mr. Ritter's statements, he's deriding the crypto establishment for ignoring the work of outsiders. My counter is not the crypto community is right to generally ignore outsiders, but that in fact they do no such thing.

: >For an example of an idea the crypto community has ignored
: >because it is truly dreadful:

: >[...]
: >: And in this way we can have hundreds or thousands of different
: >: ciphers, with more on the way all the time. That means that we can
: >: divide the worth of our information into many different ciphers, so
: >: that if any one fails, only a fraction of messages are exposed.

: >Absurdly naive. In any real project or real enterprise, the
: >same information is carried by many, many messages. The degree
: >of protection of any piece of intelligence is that of the
: >weakest of the systems carrying it.

: While that is true, that just means that, for internal encryption in an
: organization, a method should not be used that allows employees to protect
: company data with ciphers their employer does not trust.

I agree it means that, but certainly not that it "just means" that. Specifically, it should guide those employers in deciding how many ciphers to designate as trusted.

[...]
: 'Dreadful' is not the same as 'not everywhere applicable'.

True, but I'm saying that in _all_ the real projects or enterprises I know of, an attacker can gain most of the intelligence value in the message traffic by compromising only a small percentage of the messages. Are there projects in which documents do not go through many revisions? In which everyone works with a mutually exclusive subset of the information?

There is a situation worse than having all one's eggs in one basket. The problem with one basket is that there exists a potential failure that would be catastrophic. What's worse is a system in which any one of many possible failures would be catastrophic. If one accepts that in realistic applications of cryptography the same intelligence is available from many messages, then choosing from a thousand ciphers for each message moves us from one potential catastrophic

failure to many potential catastrophic failures.

--Bryan

Subject: Re: Thought question: why do public ciphers use only simple ops like shift and XOR?
Date: Sun, 25 Apr 1999 07:02:01 -0700
From: Sundial Services <info@sundialservices.com>
Message-ID: <37232059.4FA1@sundialservices.com>
References: <7fusfv$as8$1@news.umbc.edu>
Newsgroups: sci.crypt
Lines: 28

> : olson@umbc.edu (Bryan G. Olson; CMSC (G)) wrote, in part:
[...]
> : However, I don't think it's appropriate to automatically conclude that
> : everyone who expresses concern about the way in which the public
> : cryptography field is going is necessarily a crank. For example, if even a
> : layperson looks at DES, or IDEA, or SERPENT, and expresses the opinion that
> : these designs all seem too regular, too repetitious, so that some form of
> : analysis at least seems like it may be someday possible ...

I think that this is basically where -I- am coming from. If you look at the design of these Feistel ciphers, well, to me they smack of Enigma, with its clockwork-like rotation of the cipher elements which ultimately proved its downfall. Compare this to SIGABA, which with its many layers of complexity "cascading" upon one another produced what is obviously an extremely strong cipher. There is a LOT more randomness for the cryptographer to figure out.

I stare at this "more stages = more security" story and ponder if, given the extreme regularity of the cipher algorithm, this intuitive notion is actually true. Frankly, I don't believe that it is.

I see no creativity here. (So to speak!!) (So to speak!!!!)

Furthermore... the ciphers are far simpler than they need to be. A computer program can do anything. It can use as much memory as it likes. My 2,048 bit public-key could just as easily be 200K and it would be no more difficult to manage.

Subject: Re: Thought question: why do public ciphers use only simple ops like shift and XOR?
Date: Sun, 25 Apr 1999 07:04:08 -0700
From: Sundial Services <info@sundialservices.com>
Message-ID: <372320D8.59EC@sundialservices.com>
References: <37232059.4FA1@sundialservices.com>
Newsgroups: sci.crypt
Lines: 15

Sundial Services wrote:
[...]
> I think that this is basically where -I- am coming from. If you look at
> the design of these Feistel ciphers, well, to me they smack of Enigma,
> with its clockwork-like rotation of the cipher elements which ultimately
> proved its downfall. Compare this to SIGABA, which with its many layers
> of complexity "cascading" upon one another produced what is obviously an
> extremely strong cipher. There is a LOT more randomness for the
> cryptographer to figure out.

I should clarify my thought here. "The layers in SIGABA are not all the same design. The layers in an n-round Feistel cipher are, literally by definition, all the same. And all made of extremely simple primitive operations: bitwise substitution, shifting, exclusive-OR, perhaps multiplication ....

Subject: Re: Thought question: why do public ciphers use only simple ops like shift and XOR?
Date: Sun, 25 Apr 1999 15:51:42 GMT
From: m@mbsks.franken.de (Matthias Bruestle)
Message-ID: <1999Apr25.155142.3195@mbsks.franken.de>
References: <37232059.4FA1@sundialservices.com>
Newsgroups: sci.crypt
Lines: 28

Mahlzeit

Sundial Services (info@sundialservices.com) wrote:

> Furthermore... the ciphers are far simpler than they need to be. A
> computer program can do anything. It can use as much memory as it
> likes. My 2,048 bit public-key could just as easily be 200K and it
> would be no more difficult to manage.

But you wouldn't want to use this key. A 9000bit key needs about 15 minutes of a 486DX 33MHz CPU. I think the decryption/signing time raises at n^2, so a 200kbit key would require about 100 hours of this

CPU. A Pentium 200MHz, not that old, is about 10 times as fast and
would require about 10 CPU hours. Would you want to wait 10 hours
to read an email?

With all crypto applications there are speed requirements.

Mahlzeit

endergone Zwiebeltuete

--
PGP: SIG:C379A331 ENC:F47FA83D I LOVE MY PDP-11/34A, M70 and MicroVAXII!
--
Remember, even if you win the rat race -- you're still a rat.

Subject: Re: Thought question: why do public ciphers use only simple ops like shift
and XOR?
Date: Sun, 25 Apr 1999 23:49:11 -0600
From: jgfunj@vgrknf.arg (wtshaw)
Message-ID: <jgfunj-2504992349120001@dial-243-065.itexas.net>
References: <7fusfv$as8$1@news.umbc.edu>
Newsgroups: sci.crypt
Lines: 31

In article <7fusfv$as8$1@news.umbc.edu>, olson@umbc.edu (Bryan G. Olson;
CMSC (G)) wrote:

>
> There is a situation worse than having all one's eggs in one basket.
> The problem with one basket is that there exists a potential failure
> that would be catastrophic. What's worse is a system in which any
> one of many possible failures would be catastrophic. If one accepts
> that in realistic applications of cryptography the same intelligence
> is available from many messages, then choosing from a thousand
> ciphers for each message moves us from one potential catastrophic
> failure to many potential catastrophic failures.

With some effort, but it could be completely automated, using several
algorithms, it is reasonable to maximize security available not by living
in fear of the weakest algorithm but working to make sure the strongest
was included.

Consider the following key handling scheme: A OTP quality stream key is
converted to a number of complementary keys that must all be assimilated
to reestablish the real key. Those several keys are encrypted using
different algorithms. If any of the several algorithms is broken, it does
not matter because all must be broken to get at the real key.

The disadvantages are the combined length of all the keys, and needing
them all. A scheme might be devised somewhat similiar where only a
certain number of the keys would be needed. The result would be the same,
shared maximized strength of different algorithms.
--
Life's battles do not always go to the stronger of faster man...
But, sooner or later always go to the fellow who thinks he can.

Subject: Re: Thought question: why do public ciphers use only simple ops like shift
and XOR?
Date: Sat, 24 Apr 1999 01:15:17 -0600
From: jgfunj@vgrknf.arg (wtshaw)
Message-ID: <jgfunj-2404990115180001@dial-243-115.itexas.net>
References: <7fp131$dg1$1@news.umbc.edu>
Newsgroups: sci.crypt
Lines: 41

In article <7fp131$dg1$1@news.umbc.edu>, olson@umbc.edu (Bryan G. Olson;
CMSC (G)) wrote:

> Terry Ritter (ritter@io.com) wrote:
>
> [...]
> : And in this way we can have hundreds or thousands of different
> : ciphers, with more on the way all the time. That means that we can
> : divide the worth of our information into many different ciphers, so
> : that if any one fails, only a fraction of messages are exposed.
>
> Absurdly naive. In any real project or real enterprise, the
> same information is carried by many, many messages. The degree
> of protection of any piece of intelligence is that of the
> weakest of the systems carrying it.

From a herd point of view, you may be right, but specific information
between individuals is not apt to pass but once or few times at the most.
To fully follow the dialog, all parts of the conversation should be
recovered. Even when encrypted, however, the use allegory and novel in
text, security measures in themselves, should be used.
>
> : It
> : also means that *any* Opponent must keep up with new ciphers and
> : analyze and possibly break each, then design a program, or build new
> : hardware to exploit it. We can make good new ciphers cheaper than

> : they can possibly be broken. The result is that our Opponents must
> : invest far more to get far less, and this advantage does not depend
> : upon the delusion of strength which is all that cryptanalysis can
> : provide.
>
> Nonsense. The attacker just waits for the information he wants
> to be transmitted under a system he can break.
>
If certain information is so common, it may not be worth encrypting in the
first place. The idea of putting all eggs in one basket, or very few, is
not supportable; But, one should only use promising baskets in any event.
Keep 'em busy with ciphers that they have not even considered before.
--
Life's battles do not always go to the stronger of faster man...
But, sooner or later always go to the fellow who thinks he can.

Subject: Re: Thought question: why do public ciphers use only simple ops like shift
and XOR?
Date: Fri, 16 Apr 1999 12:54:41 -0500
From: Medical Electronics Lab <rosing@physiology.wisc.edu>
Message-ID: <37177961.663E@physiology.wisc.edu>
References: <371749CC.4779@sundialservices.com>
Newsgroups: sci.crypt
Lines: 47

Sundial Services wrote:
>
> When I look at most publicly-available cryptographic algorithms, I see
> that nearly all of them consist of round upon round of simple operations
> like: shift, exclusive-OR, and "bit-twiddling." Most of these ops are
> readily reversible.

Being reversible makes a cipher decipherable :-)

> About the only "original idea" I've seen, since reading discussions of
> older machines like SIGABA, is Terry Ritter's "Dynamic Substitution"
> patent. At least he is using a more complex transformation than 99.9%
> of the things I've seen ... since SIGABA ... and he's burying a lot more
> information than most designs do.

Terry's got a lot of good ideas. But even he would like a cipher
that can be analyzed completely.

> My question is, aside from possible requirements for constructing their
> ciphers in hardware, why do designers routinely limit themselves to
> these simple bitwise operators in designing ciphers? It seems to me as
> a layman that the older, more complex designs were also far more secure
> than what we have now, and that a computer program would have no
> particular difficulty implementing them. We are not building hardware
> devices; we are not limited to LFSR's.

XOR is really addition in GF(2^n) and rotation is equivelent to
multiplication by x (or squaring in a normal basis). These "simple"
operations can come from really complex math. By using math as a
basis for the creation of a cipher, you can determine the work factor
to break it more accurately.

Some of the things you want to make happen in a cipher are "avalanch"
and "diffusion". You want to make sure that if you change any one
bit in the plain text that half the bits change in the cipher text.
You also want to have a non-linear function between input and output
so there is no hope of writing down a system of equations which could
solve a cipher.

Just because something looks complex doesn't make it so. Some things
which look really simple can have very complex mathematical
relationships, and that's far more useful to a cipher design than
something which appears complex but has a simple mathematical
construction.

Patience, persistence, truth,
Dr. mike

Subject: Re: Thought question: why do public ciphers use only simple ops like shift
and XOR?
Date: Sat, 17 Apr 1999 00:16:49 -0600
From: jgfunj@vgrknf.arg (wtshaw)
Message-ID: <jgfunj-1704990016490001@dial-243-079.itexas.net>
References: <37177961.663E@physiology.wisc.edu>
Newsgroups: sci.crypt
Lines: 32

In article <37177961.663E@physiology.wisc.edu>, Medical Electronics Lab
<rosing@physiology.wisc.edu> wrote:
>
> Some of the things you want to make happen in a cipher are "avalanch"
> and "diffusion". You want to make sure that if you change any one
> bit in the plain text that half the bits change in the cipher text.
> You also want to have a non-linear function between input and output
> so there is no hope of writing down a system of equations which could
> solve a cipher.

See there, you prove his point, as avalanche paired with diffusion are essential properties of operations involving only some ciphers, and cryptography can be done with narry a bit in sight.

The design can demand so many equations be written that it is impractical to do so.

>
> Just because something looks complex doesn't make it so.

To that, I agree.

> Some things
> which look really simple can have very complex mathematical
> relationships, and that's far more useful to a cipher design than
> something which appears complex but has a simple mathematical
> construction.
>
Then there are those designs that tend to impress people because they are overly complex in construction and perhaps made so for devious purposes.
--
Too much of a good thing can be much worse than none.

Subject: Re: Thought question: why do public ciphers use only simple ops like shift and XOR?
Date: Fri, 16 Apr 1999 21:16:02 -0400
From: Boris Kazak <bkazak@worldnet.att.net>
Message-ID: <3717E0D2.225A@worldnet.att.net>
References: <371749CC.4779@sundialservices.com>
Newsgroups: sci.crypt
Lines: 56

Sundial Services wrote:
>
> When I look at most publicly-available cryptographic algorithms, I see
> that nearly all of them consist of round upon round of simple operations
> like: shift, exclusive-OR, and "bit-twiddling." Most of these ops are
> readily reversible.
>
> About the only "original idea" I've seen, since reading discussions of
> older machines like SIGABA, is Terry Ritter's "Dynamic Substitution"
> patent. At least he is using a more complex transformation than 99.9%
> of the things I've seen ... since SIGABA ... and he's burying a lot more
> information than most designs do.
>
> My question is, aside from possible requirements for constructing their
> ciphers in hardware, why do designers routinely limit themselves to
> these simple bitwise operators in designing ciphers? It seems to me as
> a layman that the older, more complex designs were also far more secure
> than what we have now, and that a computer program would have no
> particular difficulty implementing them. We are not building hardware
> devices; we are not limited to LFSR's.

As layman to layman - the most obvious reason is that these simple operations are easy to analyze. It is not by accident that the only exception to this rule is IDEA, based on modular multiplication, and this immediately brushes away a whole bunch of possible attacks.

Another observation - most published attacks against various ciphers are essentially attacking not as much the cipher per se, as its key schedule. It is not by accident that BLOWFISH is so steady, its key schedule does not provide any opportunity for related-key attacks. On the other hand, a recently published attack against IDEA makes heavy use of the fact that its subkeys are produced just by 25-bit circular shift. Use another key scheduling mechanism (same modular multiplication which is akready present in the program), and this attack will result in nothing.

As a layman, I experimented with modular multiplication mod 2^32-1 and mod 2^32+1, found the cycles produced by raising different numbers to the subsequent powers, discovered methods of testing numbers for having the multiplicative inverses, and finally wrote a program for a cipher which I call LETSIEF (FEISTEL spelled backwards). This program uses multiplication mod 2^32-1 as the combining operation between L and R halves. The speed is fantastic - multiplication mod 2^32-1 is implemented in 3 processor instructions on a Pentium, an array of 256 modular multipliers assures full plaintext dependency, inverses also occupy an array of 256 elements, so the only difference between encryption and decryption is that you take your multiplier from a conjugate array. Key scheduling uses the same multiplication routine which already exists in the program.

I am not going to post this program or to promote it in any way. It serves my purposes, I am ready to give the code to anybody who is interested, but nothing beyond that.

BTW, I also experimented with multiplication mod 2^64+1 and 2^64-1. Unfortunately, I am not so great a programmer, and my computer has no 64-bit registers. So beyond some basic knowledge, nothing yet did come into practice (but the ciphers could be terrific!).

Best wishes BNK

Subject: Re: Thought question: why do public ciphers use only simple ops like shift and XOR?
Date: Mon, 19 Apr 1999 15:38:34 -0600

From: jcoffin@taeus.com (Jerry Coffin)
Message-ID: <MPG.1182a9a3823e66899899fb@news.rmi.net>
References: <3717E0D2.225A@worldnet.att.net>
Newsgroups: sci.crypt
Lines: 33

In article <3717E0D2.225A@worldnet.att.net>, bkazak@worldnet.att.net
says...

[...]

> BTW, I also experimented with multiplication mod 2^64+1 and 2^64-1.
> Unfortunately, I am not so great a programmer, and my computer has
> no 64-bit registers. So beyond some basic knowledge, nothing yet did
> come into practice (but the ciphers could be terrific!).

...or they might not be. 2^32-1 happens to be a prime number. In
many cases, the smallest factor of your modulus has a large effect on
the security of encryption using that modulus.

By contrast, 2^64-1 is what you might call extremely composite -- its
prime factorization is (3 5 17 257 641 65537 6700417). This large
number of relatively small factors will often make this a particularly
bad choice of modulus.

Depending on what you're doing, 2^64+1 is likely to be a MUCH better
choice -- it's still not a prime, but its prime factorization is
(274177 67280421310721). In many cases, the largest prime factor is
what matters, and in this case, it's MUCH larger -- 14 digits instead
of 7 (which is also considerably larger than 2^32-1). Unfortunately,
using 2^64+1 as a modulus is likely to be fairly difficult even if you
have a 64-bit type available.

I obviously haven't studied your encryption method in detail (or at
all) so I don't _know_ that this will make a difference in your
particular case, but it's definitely something to keep in mind. Many,
many forms of encryption that work quite well in 32-bit arithmetic
basically fall to pieces when converted to use 64-bit arithmetic
instead.

Subject: Re: Thought question: why do public ciphers use only simple ops like shift
and XOR?
Date: Mon, 19 Apr 1999 19:45:17 -0400
From: Boris Kazak <bkazak@worldnet.att.net>
Message-ID: <371BC00D.3FA8@worldnet.att.net>
References: <MPG.1182a9a3823e66899899fb@news.rmi.net>
Newsgroups: sci.crypt
Lines: 53

Jerry Coffin wrote:
> ...or they might not be. 2^32-1 happens to be a prime number. In
> many cases, the smallest factor of your modulus has a large effect on
> the security of encryption using that modulus.

Sorry, 2^32-1 = 3*5*17*257*65537, but I have found nice ways to set up
key-derived multipliers in this field. The maximum length of the
multiplicative cycle is 65536, so you can select an appropriate SEED
and raise it to any power < 2^16. In fact, both the modular multiplier
and its inverse are computed in the same subroutine.
>
> By contrast, 2^64-1 is what you might call extremely composite -- its
> prime factorization is (3 5 17 257 641 65537 6700417). This large
> number of relatively small factors will often make this a particularly
> bad choice of modulus.

Also not necessarily. The important thing is the multiplicative cycle
length which can be achieved, this gives you an idea of how many
multipliers you can produce from an appropriately chosen SEED.
BTW, the only practical requirement to the SEED is that it should
produce the maximum length cycle of its powers, i.e be a generator.
>
> Depending on what you're doing, 2^64+1 is likely to be a MUCH better
> choice -- it's still not a prime, but its prime factorization is
> (274177 67280421310721). In many cases, the largest prime factor is
> what matters, and in this case, it's MUCH larger -- 14 digits instead
> of 7 (which is also considerably larger than 2^32-1). Unfortunately,
> using 2^64+1 as a modulus is likely to be fairly difficult even if you
> have a 64-bit type available.

As a matter of fact, very easy. The hex number c720a6486e45a6e2
produces in the 2^64+1 field a cycle of its own powers which is
72057331223781120 long (just under 2^56). This number is simply the
first 16 hex digits of sqrt(3), and I am sure that it will take me
not more than 15 minutes to find 5-6 numbers more like this.
(Please, don't ask me about a source code for the program, I've
written it in FORTH). So I can generate random 32-bit subkeys, raise
my SEED to these powers and I am in business... Go guess the linear
and differential properties of these multipliers, especially if they
will be chosen for encryption in a plaintext-dependent way!
>
> I obviously haven't studied your encryption method in detail (or at
> all) so I don't _know_ that this will make a difference in your
> particular case, but it's definitely something to keep in mind. Many,

> many forms of encryption that work quite well in 32-bit arithmetic
> basically fall to pieces when converted to use 64-bit arithmetic
> instead.

I do not intend to keep it secret. If you are interested (just for fun),
I am ready to discuss with you the method of file transfer
(unfortunately, I don't have a Web page).

  Thanks for your courtesy Best wishes BNK

Subject: Re: Thought question: why do public ciphers use only simple ops like shift
and XOR?
Date: Tue, 20 Apr 1999 11:42:53 -0600
From: jcoffin@taeus.com (Jerry Coffin)
Message-ID: <MPG.11866a88f9e5b896989a08@news.rmi.net>
References: <371BC00D.3FA8@worldnet.att.net>
Newsgroups: sci.crypt
Lines: 40

In article <371BC00D.3FA8@worldnet.att.net>, bkazak@worldnet.att.net
says...
> Jerry Coffin wrote:
> > ...or they might not be. 2^32-1 happens to be a prime number. In
> > many cases, the smallest factor of your modulus has a large effect on
> > the security of encryption using that modulus.
> ---
> Sorry, 2^32-1 = 3*5*17*257*65537, but I have found nice ways to set up
> key-derived multipliers in this field. The maximum length of the
> multiplicative cycle is 65536, so you can select an appropriate SEED
> and raise it to any power < 2^16. In fact, both the modular multiplier
> and its inverse are computed in the same subroutine.

Oops -- my bad. It's 2^31-1 which is a prime. Of course, if you work
in 32-bit integers, it's also 2^31-1 that you end up using as a
modulus unless you take steps to ensure against it.

However, even though I wasn't thinking very straight when posting, the
fact remains that the largest 32-bit number is a prime, and the
largest 64-bit number isn't. Interestingly enough, 2^63+1 also has a
much larger factor than 2^63-1, though it's a lot smaller than the
largest factor of 2^64+1 (only 11 digits instead of 14).

> I do not intend to keep it secret. If you are interested (just for fun),
> I am ready to discuss with you the method of file transfer
> (unfortunately, I don't have a Web page).

If it's written in Forth, I'll pass, thanks anyway. It's been many
years since the last time I tried to work in Forth at all, and from
what I remember, it's probably something that you have to either use a
lot, or you might as well forget it completely.

Then again, I suppose many people would say the same about C, C++ and
Scheme, all of which I use fairly regularly. Scheme (or almost any
LISP-like language) supports working with large integers, which tends
to be handy when you're dealing with factoring and such.

>   Thanks for your courtesy Best wishes BNK

Likewise, especially when I posted something as boneheaded as I did...

Subject: Re: Thought question: why do public ciphers use only simple ops like shift
and XOR?
Date: Tue, 20 Apr 1999 19:53:16 -0400
From: Boris Kazak <bkazak@worldnet.att.net>
Message-ID: <371D136C.411A@worldnet.att.net>
References: <MPG.11866a88f9e5b896989a08@news.rmi.net>
Newsgroups: sci.crypt
Lines: 22

Jerry Coffin wrote:
> If it's written in Forth, I'll pass, thanks anyway. It's been many
> years since the last time I tried to work in Forth at all, and from
> what I remember, it's probably something that you have to either use a
> lot, or you might as well forget it completely.

No, it's plain conventional C, even without Assembler. It is one of my
"essays" on the subject of *drunken* ciphers, where you set up a lot
of S-boxes deriving them from the key, and then encrypt using the
plaintext-dependent path through these S-boxes. so that each plaintext
will follow the maze along its own unique path. Quite entertaining...
BTW, key scheduling uses the same modular multiplication already present
in the program.
>
> Then again, I suppose many people would say the same about C, C++ and
> Scheme, all of which I use fairly regularly. Scheme (or almost any
> LISP-like language) supports working with large integers, which tends
> to be handy when you're dealing with factoring and such.
>
> >   Thanks for your courtesy Best wishes BNK
>
> Likewise, especially when I posted something as boneheaded as I did...

Subject: Re: Thought question: why do public ciphers use only simple ops like shift and XOR?
Date: Tue, 20 Apr 1999 02:50:00 GMT
From: phr@netcom.com (Paul Rubin)
Message-ID: <phrFAGvvC.3vz@netcom.com>
References: <MPG.1182a9a3823e66899899fb@news.rmi.net>
Newsgroups: sci.crypt
Lines: 10

In article <MPG.1182a9a3823e66899899fb@news.rmi.net>,
Jerry Coffin <jcoffin@taeus.com> wrote:
>...or they might not be. 2^32-1 happens to be a prime number.

```
2^32-1 = (2^16)^2-1
       = (2^16+1)*(2^16-1)
       = (2^16+1)*(2^8+1)*(2^8-1)
       = (2^16+1)*(2^8+1)*(2^4+1)*(2^4-1)
       = (2^16+1)*(2^8+1)*(2^4+1)*(2^2+1)*(2^2-1)
       = 65537   *257    *17     *5      *3
```

Subject: Re: Thought question: why do public ciphers use only simple ops like shift and XOR?
Date: Sat, 17 Apr 1999 00:03:53 -0600
From: jgfunj@vgrknf.arg (wtshaw)
Message-ID: <jgfunj-1704990003530001@dial-243-079.itexas.net>
References: <371749CC.4779@sundialservices.com>
Newsgroups: sci.crypt
Lines: 26

In article <371749CC.4779@sundialservices.com>, info@sundialservices.com wrote:

> When I look at most publicly-available cryptographic algorithms, I see
> that nearly all of them consist of round upon round of simple operations
> like: shift, exclusive-OR, and "bit-twiddling." Most of these ops are
> readily reversible.
>
> About the only "original idea" I've seen, since reading discussions of
> older machines like SIGABA, is Terry Ritter's "Dynamic Substitution"
> patent. At least he is using a more complex transformation than 99.9%
> of the things I've seen ... since SIGABA ... and he's burying a lot more
> information than most designs do.
>
> My question is, aside from possible requirements for constructing their
> ciphers in hardware, why do designers routinely limit themselves to
> these simple bitwise operators in designing ciphers? It seems to me as
> a layman that the older, more complex designs were also far more secure
> than what we have now, and that a computer program would have no
> particular difficulty implementing them. We are not building hardware
> devices; we are not limited to LFSR's.

You've got it right, cryptography is a most complicated and broad field; every cooperating to plow and plant the same furrow does not make lots of sense.
--
Too much of a good thing can be much worse than none.

Subject: Re: Thought question: why do public ciphers use only simple ops like shift and XOR?
Date: Sat, 17 Apr 1999 18:18:34 GMT
From: schneier@counterpane.com (Bruce Schneier)
Message-ID: <3718cff6.15699939@news.visi.com>
References: <371749CC.4779@sundialservices.com>
Newsgroups: sci.crypt
Lines: 25

On Fri, 16 Apr 1999 07:31:40 -0700, Sundial Services
<info@sundialservices.com> wrote:

>When I look at most publicly-available cryptographic algorithms, I see
>that nearly all of them consist of round upon round of simple operations
>like: shift, exclusive-OR, and "bit-twiddling." Most of these ops are
>readily reversible.

Operations from the RISC subset are efficient on a wide variety of microprocessors. Look at the AES submissions. Algorithms that limited themselves to those operations--Serpent, Rijndael, Twofish--had realtively equivalent performance on 8-bit CPUs, 32-bit CPUs, smart card, DSPs, etc. Algorithms that used more complicated operations like data dependent rotations and multiplications--Mars, RC6, DFC--had widely different performance depending on the particular characteristics of the CPU it is running on.

For a standard cipher at least, sticking to the RISC subset is just smart.

Bruce
****
Bruce Schneier, President, Counterpane Systems Phone: 612-823-1098
101 E Minnehaha Parkway, Minneapolis, MN 55419 Fax: 612-823-1590
Free crypto newsletter. See: http://www.counterpane.com

Subject: Re: Thought question: why do public ciphers use only simple ops like shift and XOR?
Date: 18 Apr 99 02:10:22 GMT
From: jsavard@ecn.ab.ca ()
Message-ID: <37193f0e.0@ecn.ab.ca>
References: <3718cff6.15699939@news.visi.com>
Newsgroups: sci.crypt
Lines: 15

Bruce Schneier (schneier@counterpane.com) wrote:
: For a standard cipher at least, sticking to the RISC subset is just
: smart.

My comment on that paragraph is that he forgot S-boxes, which, if one is using the RISC subset, one cannot omit. But looking at the rest of his post, I don't think he was thinking of things like data-dependent rotations, multiplication, and so on, as much as he was thinking of more creative use of S-boxes or more creative combinations of RISC-subset operations.

Think FROG. Or recall my "Mishmash" posting. This, I think, is the kind of thing he is talking about.

John Savard

Subject: Re: Thought question: why do public ciphers use only simple ops like shift and XOR?
Date: Sun, 18 Apr 1999 00:41:15 -0600
From: jgfunj@vgrknf.arg (wtshaw)
Message-ID: <jgfunj-1804990041160001@dial-243-094.itexas.net>
References: <3718A7C9.12B5EEF@null.net>
<3718324d.13916819@news.io.com>
Newsgroups: sci.crypt
Lines: 15

In article <3718A7C9.12B5EEF@null.net>, "Douglas A. Gwyn"
<DAGwyn@null.net> wrote:
>
> I guess you're talking about AES. If time constraints allow,
> that would be one reasonable part of the evaluation procedure,
> but you still have to drawn the line somewhere and pick the
> best-to-date.

Ah, elections do come up at some point. As I remember, the final pick is to be submitted to higher, political, authority for *approval*, which is apt not to be a technical decision based on purely scientific considerations. Meanwhile, back at the ranch, we can make things better by trying to go beyond such a seal.
--
Too much of a good thing can be much worse than none.

Subject: Re: Thought question: why do public ciphers use only simple ops like shift and XOR?
Date: Sun, 18 Apr 1999 00:45:35 -0600
From: jgfunj@vgrknf.arg (wtshaw)
Message-ID: <jgfunj-1804990045350001@dial-243-094.itexas.net>
References: <3718A84E.A90A3130@null.net>
<jgfunj-1704990016490001@dial-243-079.itexas.net>
Newsgroups: sci.crypt
Lines: 15

In article <3718A84E.A90A3130@null.net>, "Douglas A. Gwyn"
<DAGwyn@null.net> wrote:

> wtshaw wrote:
> > The design can demand so many equations be written that it is
> > impractical to do so.
>
> How could the design be conveyed to the implementor, then?

I was thinking more in terms of a simple design, guess, where the burden of writing the equations would be on the attacker who would be trying to make sense out of lots of ciphertext. You know, something easy to do knowing the key, and impractical not knowing it.
--
Too much of a good thing can be much worse than none.

Subject: Re: Thought question: why do public ciphers use only simple ops like shift and XOR?
Date: Sun, 18 Apr 1999 23:36:51 -0600
From: jgfunj@vgrknf.arg (wtshaw)
Message-ID: <jgfunj-1804992336510001@dial-243-098.itexas.net>
References: <3719F8DA.B280DB30@null.net>
<jgfunj-1804990041160001@dial-243-094.itexas.net>
Newsgroups: sci.crypt
Lines: 28

In article <3719F8DA.B280DB30@null.net>, "Douglas A. Gwyn"
<DAGwyn@null.net> wrote:

> wtshaw wrote:
> > Ah, elections do come up at some point. As I remember, the final
> > pick is to be submitted to higher, political, authority for
> > *approval*, which is apt not to be a technical decision ...
>
> The technical decision would already have been made, and any
> further process would be simply an approve/disapprove decision.

That is an easy prediction for a technical person. In politics, the rule
is there are no rules, except that rules of more equal for those who
contribute to the right people.
>
> I don't know what "elections" have to do with it. You can't
> think that the electorate in general cares one whit about AES.

End decisions are to be made by political cronies. My reference to
elections suggests that we should all get involved in the political level
as well if we want the best AES, such that it is, to come forward.
Getting involved means getting closer to actual candidates and
spotlighting the various crypto related issues to them. You can't be just
a technician any more and be responsible.
--
A new random permutation generator: You put X windoze
machines in a room, merely start them up, and record the
order in which they eventually crash on their own.

Subject: Re: Thought question: why do public ciphers use only simple ops like shift
and XOR?
Date: 17 Apr 1999 03:25:46 GMT
From: David A Molnar <dmolnar@fas.harvard.edu>
Message-ID: <7f8uvq$h94$1@news.fas.harvard.edu>
References: <3718235E.CB681D9C@null.net>
   <3717ba72.20758328@news.io.com>
Newsgroups: sci.crypt
Lines: 51

Douglas A. Gwyn <DAGwyn@null.net> wrote:
> The only valid thing they could say is that they don't know any
> way to demonstrate that a cipher is inherently secure (to some
> agreed level of confidence, under ideal operating conditions).
> However, there *have* been a few academic publications purporting
> to demonstrate provable security for certain systems. A valid
> proof would mean that the system was secure so long as nothing
> went wrong. (That is always an important practical caveat, since
> things do occasionally go wrong.)

Could you mention the publications of which you're thinking? Most of the
provable security I've seen comes in terms of statements like "if this
scheme is breakable, then factoring integers is easy," usually by providing
an argument that any adversary who magically comes across the key
necessarily gains enough information to do something ridiculous like
factor huge numbers. I'm just wondering if that is to what you're referring.

In this vein, right now I really like the papers by Mihir Bellare and
Phil Rogaway which advocate and demonstrate what they call "exact security"
-- giving the amount of time and the probability of breaking a scheme
in terms of values like "amount of computing power posessed by adversary",
"number of {chosen | known | adaptive chosen } ciphertexts known", and
so on. There's an overview at
http://www-cse.ucsd.edu/users/mihir/papers/pops.ps
which sort of falls into the category of not quite bragging about the
acheivement.

One of the nice things about this area is that the definition of what
it means to be "secure" can be made precise and formal. That doesn't prove
anything by itself, but offers a way to start proving things. This
is rather fun, if you can swallow the assumptions
(e.g. RSA is hard).

Anyway, PKCS #1 v2.0 and lots of the upcoming IEEE P1363 standard will
be based on this problem. So, while I'm not quite sure what 'a few' means
yet, it is important to know about. If only to see if and how it applies to the
standard sci.crypt discussion on the stength of cipher X, or the best
cipher for application Y.

If that's not quite what you meant -- for instance if you mean tools like
code specification, hardening card protocols against attacks on the device,
and so on, then I'd be interested in seeing 'em.

The real problem is as you say -- prolly best summed up by Bruce Schneier's
pronouncement along thie lines of "The math is perfect, the hardware is
not so great, the software is corrupt, and the people are horrible) . That,
and sometimes stupid assumptions. seem to be worth watching the **** out
for when implementing this kind of thing...

-David

shift and XOR?

Subject: Re: Thought question: why do public ciphers use only simple ops like
Date: Sat, 17 Apr 1999 04:39:52 GMT
From: "Douglas A. Gwyn" <DAGwyn@null.net>
Message-ID: <37181E71.17709C61@null.net>
References: <371749CC.4779@sundialservices.com>
Newsgroups: sci.crypt
Lines: 43

Sundial Services wrote:
> like: shift, exclusive-OR, and "bit-twiddling." Most of these ops are
> readily reversible.

Yes, that ensures that decryption is feasible.

> About the only "original idea" I've seen, since reading discussions of
> older machines like SIGABA, is Terry Ritter's "Dynamic Substitution"
> patent. At least he is using a more complex transformation than 99.9%
> of the things I've seen ... since SIGABA ... and he's burying a lot more
> information than most designs do.

Complexity in itself is no guarantee of security; witness Knuth's
"super-random" number generator (Algorithm K). As to how "deeply
buried" the information is, how do you determine that? Is there
some computable figure of merit, or what?

> My question is, aside from possible requirements for constructing their
> ciphers in hardware, why do designers routinely limit themselves to
> these simple bitwise operators in designing ciphers?

Simpler systems are, usually, easier to analyze more thoroughly.
The more thoroughly we understand a class of systems, the more
confident we can be that other analysts won't find some shortcut.

> It seems to me as a layman that the older, more complex designs
> were also far more secure than what we have now,

How do you know what we have now? The public didn't have access
to SIGABA systems back then, just as they don't have access to
<censored> today.

> and that a computer program would have no particular difficulty
> implementing them. We are not building hardware devices; we are
> not limited to LFSR's.

It is true that simulation of a Hagelin or Hebern machine, or SIGABA,
is easy these days, and that computer programs don't *have* to follow
a classical hardware model. However, things like LFSRs have been
thoroughly studied by cryptomathematicians, so informed decisions
can be made about how (or whether) to use them. If you attempt a
new system structure, until it is well understood mathematically,
you'd have no justification for thinking it to be secure.

shift and XOR?

Subject: Re: Thought question: why do public ciphers use only simple ops like
Date: Sat, 17 Apr 1999 05:00:53 GMT
From: "Douglas A. Gwyn" <DAGwyn@null.net>
Message-ID: <3718235E.CB681D9C@null.net>
References: <3717ba72.20758328@news.io.com>
Newsgroups: sci.crypt
Lines: 52

Terry Ritter wrote:
> >But, If I learn to break the ciphers of others and
> >use my experience to create a new cipher that others cannot break it will be
> >listened to because I am known to be knowledgeable in how ciphers work.
> Nonsense. Knowing how to break some ciphers does not mean that you
> know how ciphers work. ...

I think the truth is somewhere in between. I myself maintain that
if you know too little about how cryptosystems are broken, you also
don't know all the potential vulnerabilities of a system you may
design, and so unless you have unusual "beginner's luck", your
system is bound to be vulnerable. Worse, it is vulnerable in ways
that were *preventable* if only you hadn't tried to take a shortcut
to success...

> We only know what success is reported in the academic literature.
> Unfortunately, when we use a cipher, we are very rarely concerned
> whether academics can break our cipher or not. We are instead
> concerned about "bad guys," and they don't tell us when they have been
> successful.

That is the reason for "tiger teams", who act the part of bad guys.
If your system hasn't been attacked by cryptanalysts who know *how*
to mount such an attack, then it hasn't undergone sufficient Quality
Control.

> >... Schneier and others have acknowledged that any cipher can be
> >broken at any time.

The only valid thing they could say is that they don't know any
way to demonstrate that a cipher is inherently secure (to some
agreed level of confidence, under ideal operating conditions).

However, there *have* been a few academic publications purporting
to demonstrate provable security for certain systems. A valid
proof would mean that the system was secure so long as nothing
went wrong. (That is always an important practical caveat, since
things do occasionally go wrong.)

Absence of knowledge is not knowledge of absence.

> ... He would thus have us believe that the lack of
> information about weakness in one cipher is superior to
> information of impractical weakness in another cipher.

The problem is, a decision has to be made, despite having
incomplete information. All other things being equal, a
demonstrated weakness is *some* evidence against that
system, even if we can't quantify how much, which would
tip the balance. But when there are factor both pro and
con, then your criticism is apropos -- we need to know
the relative amount of weight to give each factor if we
want to make the most rational decision.

Subject: Re: Thought question: why do public ciphers use only simple ops like
shift and XOR?
Date: Sat, 17 Apr 1999 07:03:48 GMT
From: ritter@io.com (Terry Ritter)
Message-ID: <3718324d.13916819@news.io.com>
References: <3718235E.CB681D9C@null.net>
Newsgroups: sci.crypt
Lines: 91

On Sat, 17 Apr 1999 05:00:53 GMT, in <3718235E.CB681D9C@null.net>, in
sci.crypt "Douglas A. Gwyn" <DAGwyn@null.net> wrote:

>Terry Ritter wrote:
>> >But, If I learn to break the ciphers of others and
>> >use my experience to create a new cipher that others cannot break it will be
>> >listened to because I am known to be knowledgeable in how ciphers work.
>> Nonsense. Knowing how to break some ciphers does not mean that you
>> know how ciphers work. ...
>
>I think the truth is somewhere in between. I myself maintain that
>if you know too little about how cryptosystems are broken, you also
>don't know all the potential vulnerabilities of a system you may
>design, and so unless you have unusual "beginner's luck", your
>system is bound to be vulnerable. Worse, it is vulnerable in ways
>that were *preventable* if only you hadn't tried to take a shortcut
>to success...

I agree with this to some extent. In particular, I have experienced
being "blind" to particular attacks which others have seen, so I am
not sure that a person can expect to be all things in this process. I
would like to see cryptanalysis be more open to changes in the design.
Currently, cryptanalysis seems to be some sort of "one shot" contest
against the cryptographer, as opposed to an interactive joint process
to attain a better cipher.

>> We only know what success is reported in the academic literature.
>> Unfortunately, when we use a cipher, we are very rarely concerned
>> whether academics can break our cipher or not. We are instead
>> concerned about "bad guys," and they don't tell us when they have been
>> successful.
>
>That is the reason for "tiger teams", who act the part of bad guys.
>If your system hasn't been attacked by cryptanalysts who know *how*
>to mount such an attack, then it hasn't undergone sufficient Quality
>Control.

Even so, we still don't know that their guys aren't better, or even
just luckier. I think it sometimes just takes a particular point of
view to enable an alternative -- possibly much easier -- attack. And
it may not be the smartest guy who has that new point of view.

>> >... Schneier and others have acknowledged that any cipher can be
>> >broken at any time.
>
>The only valid thing they could say is that they don't know any
>way to demonstrate that a cipher is inherently secure (to some
>agreed level of confidence, under ideal operating conditions).
>However, there *have* been a few academic publications purporting
>to demonstrate provable security for certain systems. A valid
>proof would mean that the system was secure so long as nothing
>went wrong. (That is always an important practical caveat, since
>things do occasionally go wrong.)

I don't have a problem with assumptions that nothing will go wrong
(although I would expect a real design to consider this). But all of
the proofs I have seen either imply a very heavy computational burden
or have made very significant assumptions simply to get the proof.
Such results may be useful when and if we can prove the basic
assumption, but I am unaware that we can.

>Absence of knowledge is not knowledge of absence.
>
>> ... He would thus have us believe that the lack of
>> information about weakness in one cipher is superior to
>> information of impractical weakness in another cipher.
>
>The problem is, a decision has to be made, despite having
>incomplete information. All other things being equal, a
>demonstrated weakness is *some* evidence against that
>system, even if we can't quantify how much, which would
>tip the balance. But when there are factor both pro and
>con, then your criticism is apropos -- we need to know
>the relative amount of weight to give each factor if we
>want to make the most rational decision.

That seems like a reasonable position.

I guess I would suggest that if the goal was to get the best cipher,
we would see a post-analysis re-design phase intended to fix known
problems, with the final comparison being made between full-strength
designs.

Terry Ritter ritter@io.com http://www.io.com/~ritter/
Crypto Glossary http://www.io.com/~ritter/GLOSSARY.HTM

Subject: Re: Thought question: why do public ciphers use only simple ops like shift
and XOR?
Date: Sun, 18 Apr 1999 00:18:55 -0600
From: jgfunj@vgrknf.arg (wtshaw)
Message-ID: <jgfunj-1804990018560001@dial-243-094.itexas.net>
References: <3718324d.13916819@news.io.com>
Newsgroups: sci.crypt
Lines: 24

In article <3718324d.13916819@news.io.com>, ritter@io.com (Terry Ritter) wrote:
>
> I guess I would suggest that if the goal was to get the best cipher,
> we would see a post-analysis re-design phase intended to fix known
> problems, with the final comparison being made between full-strength
> designs.

Even some of the worst overall ciphers that have come to light here and
there can have a novelity within which should be cherished by the crypto
community.

It would seem that the best use of the talent involved in the AES process
would be to identify any new techniques that might have crept in, and see
how these could be used to supplement the best of whatelse we know. To
stop with what we have seen thusfar would be a big mistake. It is in the
interest of cryptography for some sort of evolutionary recombination to
continue, with active support of those civilians involved, which is almost
a given as the process is merely a seed for growing beyond any government
imposed limits.

Well known are some of my reservations about AES, but I cheer on those who
even try to do the best they can, even with self-imposed handicaps.
--
Too much of a good thing can be much worse than none.

Subject: Re: Thought question: why do public ciphers use only simple ops like shift
and XOR?
Date: Sat, 17 Apr 1999 14:28:05 GMT
From: "Douglas A. Gwyn" <DAGwyn@null.net>
Message-ID: <3718A84E.A90A3130@null.net>
References: <jgfunj-1704990016490001@dial-243-079.itexas.net>
Newsgroups: sci.crypt
Lines: 5

wtshaw wrote:
> The design can demand so many equations be written that it is
> impractical to do so.

How could the design be conveyed to the implementor, then?

Subject: Re: Thought question: why do public ciphers use only simple ops like shift
and XOR?
Date: Sun, 18 Apr 1999 13:24:01 GMT
From: "Douglas A. Gwyn" <DAGwyn@null.net>
Message-ID: <3719F8DA.B280DB30@null.net>
References: <jgfunj-1804990041160001@dial-243-094.itexas.net>
Newsgroups: sci.crypt
Lines: 10

wtshaw wrote:
> Ah, elections do come up at some point. As I remember, the final
> pick is to be submitted to higher, political, authority for
> *approval*, which is apt not to be a technical decision ...

The technical decision would already have been made, and any
further process would be simply an approve/disapprove decision.

I don't know what "elections" have to do with it. You can't
think that the electorate in general cares one whit about AES.

Subject: Re: Thought question: why do public ciphers use only simple ops like shift
and XOR?
Date: Sun, 18 Apr 1999 15:37:54 +0200
From: "H. Ellenberger" <hansell@smile.ch>
Message-ID: <3719E032.5E53C5AF@smile.ch>
References: <37179b67.12809750@news.io.com>
Newsgroups: sci.crypt
Lines: 22

Terry Ritter wrote:

> >[...]
>
> The truth is that we *never* know the "real" strength of a cipher. No
> matter how much review or cryptanalysis a cipher gets, we only have
> the latest "upper bound" for strength. The lower bound is zero: Any
> cipher can fail at any time.

Correct, however you only describe the bewildering lack of a sound
theoretical foundation of the subject matter.

> Since we have only an upper bound for the strength of any cipher, any
> confidence we may have is no more than our own delusion. We wish and
> hope for cipher strength, and -- absent a specific proof otherwise --
> we gradually come to believe in it. But that does not make it true.

Correct. Should we therfore stop analyzing existing ciphers?

Subject: Re: Thought question: why do public ciphers use only simple ops like
shift and XOR?
Date: Sun, 18 Apr 1999 19:27:29 GMT
From: ritter@io.com (Terry Ritter)
Message-ID: <371a31eb.2079148@news.io.com>
References: <3719E032.5E53C5AF@smile.ch>
Newsgroups: sci.crypt
Lines: 36

On Sun, 18 Apr 1999 15:37:54 +0200, in <3719E032.5E53C5AF@smile.ch>,
in sci.crypt "H. Ellenberger" <hansell@smile.ch> wrote:

>Terry Ritter wrote:
>
>> >[...]
>>
>> The truth is that we *never* know the "real" strength of a cipher. No
>> matter how much review or cryptanalysis a cipher gets, we only have
>> the latest "upper bound" for strength. The lower bound is zero: Any
>> cipher can fail at any time.
>
>Correct, however you only describe the bewildering lack of a sound
>theoretical foundation of the subject matter.

Incorrect. The problem is both theoretical and practical; there is
ample reason to make serious changes. And, I do not *only* describe
the problem, but have also given prescriptions for increasing system
strength even when every cipher is suspect.

>> Since we have only an upper bound for the strength of any cipher, any
>> confidence we may have is no more than our own delusion. We wish and
>> hope for cipher strength, and -- absent a specific proof otherwise --
>> we gradually come to believe in it. But that does not make it true.
>
>Correct. Should we therfore stop analyzing existing ciphers?

No. We should stop depending on any small set of ciphers, no matter
how well analyzed.

Terry Ritter ritter@io.com http://www.io.com/~ritter/
Crypto Glossary http://www.io.com/~ritter/GLOSSARY.HTM

Subject: Re: Thought question: why do public ciphers use only simple ops like shift
and XOR?
Date: Sat, 17 Apr 1999 13:01:26 GMT
From: dscott@networkusa.net
Message-ID: <7fa0n5$v4m$1@nnrp1.dejanews.com>
References: <371749CC.4779@sundialservices.com>
Newsgroups: sci.crypt
Lines: 47

In article <371749CC.4779@sundialservices.com>,
  info@sundialservices.com wrote:

> When I look at most publicly-available cryptographic algorithms, I see
> that nearly all of them consist of round upon round of simple operations
> like: shift, exclusive-OR, and "bit-twiddling." Most of these ops are
> readily reversible.
>

  One really is not left with much choice in that if one does not use reversible operations one may end up doing hashing which is not encryption. But you are right that most encryption methods use operations that are readily reversible. That is one reason I use as large an S-table that is possible in my encryption programs. Most people complain that my S-tables are to large. Since the size of s-tables in scott18u.zip is effectively larger than a one million byte key.

> About the only "original idea" I've seen, since reading discussions of
> older machines like SIGABA, is Terry Ritter's "Dynamic Substitution"
> patent. At least he is using a more complex transformation than 99.9%
> of the things I've seen ... since SIGABA ... and he's burying a lot more
> information than most designs do.
>

  If you want to see originality look at scott19u.zip

> My question is, aside from possible requirements for constructing their
> ciphers in hardware, why do designers routinely limit themselves to
> these simple bitwise operators in designing ciphers? It seems to me as
> a layman that the older, more complex designs were also far more secure
> than what we have now, and that a computer program would have no
> particular difficulty implementing them. We are not building hardware
> devices; we are not limited to LFSR's.
>
--

  My feeling is that the method that computers use should involve much more operations than what the public is use to. My code treats the whole file as a block. Which is something else the current blessed methods do not do.

David Scott

http://cryptography.org/cgi-bin/crypto.cgi/Misc/scott19u.zip
http://members.xoom.com/ecil/index.htm

----------== Posted via Deja News, The Discussion Network ==----------
http://www.dejanews.com/ Search, Read, Discuss, or Start Your Own

---

Subject: Re: Thought question: why do public ciphers use only simple ops like shift and XOR?
Date: Sun, 18 Apr 1999 00:35:24 -0600
From: jgfunj@vgrknf.arg (wtshaw)
Message-ID: <jgfunj-1804990035240001@dial-243-094.itexas.net>
References: <7fa0n5$v4m$1@nnrp1.dejanews.com>
Newsgroups: sci.crypt
Lines: 14

In article <7fa0n5$v4m$1@nnrp1.dejanews.com>, dscott@networkusa.net wrote:
>
>   My feeling is that the method that computers use should involve much
> more operations than what the public is use to. My code treats the
> whole file as a block. Which is something else the current blessed
> methods do not do.
>
Which is strange since the one thing that you highlight is something that perhaps is the biggest weakness in the utility of what you have done.
And, trying to make the various blessed methods somehow stronger by making block affect each other through shoddy methods is also a step in the wrong direction. All-or-nothing logic has always been considered as a fallacy.
--
Too much of a good thing can be much worse than none.

Subject: Re: Thought question: why do public ciphers use only simple ops like shift and XOR?
Date: Sun, 18 Apr 1999 13:38:07 GMT
From: dscott@networkusa.net
Message-ID: <7fcn7v$3nv$1@nnrp1.dejanews.com>
References: <jgfunj-1804990035240001@dial-243-094.itexas.net>
Newsgroups: sci.crypt
Lines: 34

In article <jgfunj-1804990035240001@dial-243-094.itexas.net>,
  jgfunj@vgrknf.arg (wtshaw) wrote:
> In article <7fa0n5$v4m$1@nnrp1.dejanews.com>, dscott@networkusa.net wrote:
> >
> >   My feeling is that the method that computers use should involve much
> > more operations than what the public is use to. My code treats the
> > whole file as a block. Which is something else the current blessed
> > methods do not do.
> >
> Which is strange since the one thing that you highlight is something that
> perhaps is the biggest weakness in the utility of what you have done.
> And, trying to make the various blessed methods somehow stronger by making

> block affect each other through shoddy methods is also a step in the wrong
> direction. All-or-nothing logic has always been considered as a fallacy.
> --

  Since when has the "All-or-nothing logic" been considered as a fallacy
or what have you been smoking. I am sure that would be news to Mr R. of RSA
fame. All or nothing encryption has not been around very long and is not
something groups like NSA are very ready to deal with. I would like to see
some so called expert say that "wrapped PCBC" is a step in the wrong direction
making the socalled blessed ciphers weaker. Quite the opossite is true
since wrapped PCBC would add the "all or nothing logic" to the AES
candidates as well as solve the problem of handling files or sub files
that are not made up of a number of bits that are a muliply of the block
cipher size used.

David A. Scott

http://cryptography.org/cgi-bin/crypto.cgi/Misc/scott19u.zip
http://members.xoom.com/ecil/index.htm

----------== Posted via Deja News, The Discussion Network ==----------
http://www.dejanews.com/ Search, Read, Discuss, or Start Your Own

---

Subject: Re: Thought question: why do public ciphers use only simple ops like shift
and XOR?
Date: Sun, 18 Apr 1999 23:50:41 -0600
From: jgfunj@vgrknf.arg (wtshaw)
Message-ID: <jgfunj-1804992350410001@dial-243-098.itexas.net>
References: <7fcn7v$3nv$1@nnrp1.dejanews.com>
Newsgroups: sci.crypt
Lines: 32

In article <7fcn7v$3nv$1@nnrp1.dejanews.com>, dscott@networkusa.net wrote:
>
> Since when has the "All-or-nothing logic" been considered as a fallacy
> or what have you been smoking.

It's bigger than mere crypto, it applies to all logic. Formally, the
concept was presented to me eons ago in Freshman English, but I was
already familiar with the concept, with the others that go along with it.

It's also put in the wisdom of not putting all your eggs in one basket,
which is rather a historically known truth.

> I am sure that would be news to Mr R. of RSA fame.

He surely is a nice fellow, but I am also pretty sure he is liberally
educated in basic logic as well.

> All or nothing encryption has not been around very long and is not
> something groups like NSA are very ready to deal with.

I doubt that the methods used by many are that much of a problem. I will
grant that you do something better there.

I consider the technology just a patch that can be used on any of several
weaker-than- they-should-be algorithms, not to consider yours in that
lower class at all. However, the practice limits the use of your
algorithms. There may be a fix to what you have done, making your
encryption more acceptable to me, but you may not like the idea at first.
--
A new random permutation generator: You put X windoze
machines in a room, merely start them up, and record the
order in which they eventually crash on their own.

Subject: Re: Thought question: why do public ciphers use only simple ops like shift
and XOR?
Date: Mon, 19 Apr 1999 02:24:57 GMT
From: dianelos@tecapro.com
Message-ID: <7fe45i$730$1@nnrp1.dejanews.com>
References: <371749CC.4779@sundialservices.com>
Newsgroups: sci.crypt
Lines: 40

In article <371749CC.4779@sundialservices.com>,
  info@sundialservices.com wrote:

> When I look at most publicly-available cryptographic algorithms, I see
> that nearly all of them consist of round upon round of simple operations
> like: shift, exclusive-OR, and "bit-twiddling
> ...
> My question is, aside from possible requirements for constructing their
> ciphers in hardware, why do designers routinely limit themselves to
> these simple bitwise operators in designing ciphers? It seems to me as
> a layman that the older, more complex designs were also far more secure
> than what we have now, and that a computer program would have no
> particular difficulty implementing them. We are not building hardware
> devices; we are not limited to LFSR's.
>

    Whether you use simple or complex operations to describe a cipher

is not relevant: a multiplication can be seen as a sequence of SHIFTs and ADDs; in fact any cipher can in principle be expressed using only ANDs and NOTs. What is relevant in practice is to have a cipher design with fast software execution on some hardware platform. Therefore it can be useful to use the more complex machine instructions. For example a multiplication, when available in hardware, is much faster than the corresponding sequence of SHIFTs and ADDs. The designer will always try to make the hardware processor expend as much cryptographically useful work as possible within a particular time frame. That is why it is so difficult to design a cipher that is fast on many different processor designs. In other words, if a cipher is especially optimized for one platform it will probably be comparatively slower on others.

DES, the mother of all ciphers, uses only XORs, substitutions, and bit transpositions. When I designed Frog, I decided to use only XORs and substitutions (and continue the tradition), even though I knew that ADD has better diffusion properties than XOR. In all synchronous processors ADD takes the same time as XOR and I think now that I made a bad decision then.

----------== Posted via Deja News, The Discussion Network ==----------
http://www.dejanews.com/ Search, Read, Discuss, or Start Your Own

---

Subject: Re: Thought question: why do public ciphers use only simple ops like shift and XOR?
Date: Mon, 19 Apr 1999 22:05:58 GMT
From: jsavard@tenMAPSONeerf.edmonton.ab.ca (John Savard)
Message-ID: <371ba7fd.23385300@news.prosurfr.com>
References: <7fe45i$730$1@nnrp1.dejanews.com>
Newsgroups: sci.crypt
Lines: 21

dianelos@tecapro.com wrote, in part:

> DES, the mother of all ciphers, uses only XORs, substitutions, and
> bit transpositions. When I designed Frog, I decided to use only
> XORs and substitutions (and continue the tradition), even though I
> knew that ADD has better diffusion properties than XOR. In all
> synchronous processors ADD takes the same time as XOR and I think
> now that I made a bad decision then.

While at first the poster does appear to be making the important mistake of forgetting that most block ciphers do include one complex operation - the S-box - reading the post further, and noting the examples he used, such as SIGABA, led me to the conclusion that he wasn't really asking about why not many block ciphers use, say, multiply instructions, but instead was asking why they don't use more involved structures with a little creativity.

In other words, I think he was asking why block ciphers weren't more like FROG! (My original reply to his post goes into this at greater length.)

John Savard (teenerf<-)
http://members.xoom.com/quadibloc/index.html

Subject: Re: Thought question: why do public ciphers use only simple ops like shift and XOR?
Date: Wed, 21 Apr 1999 11:46:08 GMT
From: SCOTT19U.ZIP_GUY <dscott@networkusa.net>
Message-ID: <7fkdq0qra1@nnrp1.dejanews.com>
References: <3718AA8A.B2B460DE@null.net>
<7fa0n5$v4m$1@nnrp1.dejanews.com>
Newsgroups: sci.crypt
Lines: 45

In article <3718AA8A.B2B460DE@null.net>,
"Douglas A. Gwyn" <DAGwyn@null.net> wrote:
> dscott@networkusa.net wrote:
> > ... Most people complain that my S-tables are too large. Since the
> > size of s-tables in scott18u.zip is effectively larger than a one
> > million byte key.
>
> I think the question in their minds is, whether such a large S-table
> is *essential* to attain the desired level of security, or whether
> other approaches might be comparably secure.
>
> > My feeling is that the method that computers use should involve much
> > more operations than what the public is use to. My code treats the
> > whole file as a block. Which is something else the current blessed
> > methods do not do.
>
> The immediate thought one has is that treating the whole message
> would be a problem in many applications where data is generated
> in a stream, for example a TELNET session. However, a whole-file
> method can be applied to small blocks simply by treating each
> block in the sequence as a separate file. That would lose any
> benefit from the large-file property, however. Have you any
> estimate of the security of your method when used with small
> "files" (say, 512 bytes or less)?
>

Well from reading Ritters stuff one can not really estimate

the security. However it would treat the 512 bytes as a single
block. I think if one used a fixed block size of any size say
512 bytes. Then it would be easyer to study the cipher for its
weak points. But it is not the fastest method out there. Many
seem concerned with speed. I feel in my gut that the faster a
method is then there is likely an easy method to break it.
My methods reqire a lot of time compared to others. But I feel
if one wants real security then my methods are the way to go.

David A. Scott
--
http://cryptography.org/cgi-bin/crypto.cgi/Misc/scott19u.zip
http://members.xoom.com/ecil/index.htm
NOTE EMAIL address is for SPAMERS
to email me use address on WEB PAGE

----------== Posted via Deja News, The Discussion Network ==----------
http://www.dejanews.com/ Search, Read, Discuss, or Start Your Own

---

Subject: Re: Thought question: why do public ciphers use only simple ops like shift
and XOR?
Date: Thu, 22 Apr 1999 02:15:17 GMT
From: tomstdenis@my-dejanews.com
Message-ID: <7fm0ni$be4$1@nnrp1.dejanews.com>
References: <371e2003.6163199@news.io.com>
Newsgroups: sci.crypt
Lines: 50

-----BEGIN PGP SIGNED MESSAGE-----
Hash: SHA1

> I find "rounds" (the repeated application of the same operation)
silly
> and I don't use them. I do use "layers" in which different
operations
> are applied in each layer.
>
> And I think that making a cipher more difficult to analyze can only
> benefit the Opponents who have more resources for analysis.
> Personally, I try to make ciphers as conceptually *simple* as
possible
> (though not simpler). Simple does not mean weak; simple means
> appropriately decomposing the cipher into relatively few types of
> substantial subcomponent which can be understood on their own, then
> using those components in clear, structured ways.

This is true. While you could encoporate a variety of P and S
functions
in one cipher. If you had for example 24 rounds you could have 6
different P and S functions, each with 4 stagered rounds each.

I also believe in keeping ciphers simple. In some cases difficult
just comes
with it (public-key).

I would suggest if anyone is getting started to read about the
following
ciphers, which are easy to read, implement and study.

IDEA, Blowfish, RC5, TEA, X-TEA and CAST.

Tom
-----BEGIN PGP SIGNATURE-----
Version: PGPfreeware 6.0.2i

iQA/AwUBNx6E28nv2fqXBZQeEQILNgCdHThETQtVxpZoKLTRPx5nbuz8Vw8AoNDO
kG/DtwpLc1oyT5c8xOWwmg3Q
=8iVV
-----END PGP SIGNATURE-----

--
PGP public keys. SPARE key is for daily work, WORK key is for
published work. The spare is at
'http://members.tripod.com/~tomstdenis/key_s.pgp'. Work key is at
'http://members.tripod.com/~tomstdenis/key.pgp'. Try SPARE first!

----------== Posted via Deja News, The Discussion Network ==----------
http://www.dejanews.com/ Search, Read, Discuss, or Start Your Own

---

Subject: Re: Thought question: why do public ciphers use only simple ops like shift
and XOR?
Date: Thu, 22 Apr 1999 13:50:15 GMT
From: dianelos@tecapro.com
Message-ID: <7fn9el$e2v$1@nnrp1.dejanews.com>
References: <371a56a8.198396@news.prosurfr.com>
Newsgroups: sci.crypt
Lines: 54

In article <371a56a8.198396@news.prosurfr.com>,
  jsavard@tenMAPSONeerf.edmonton.ab.ca (John Savard) wrote:
> ...

>Two comments are warranted here.
>
> >- Since cryptanalysis represents the "hard" part of the work in designing
> >a cipher, this is why cipher designers should themselves know something
> >about cryptanalysis;
>
> >- And I think you can see why this design process actually _increases_ the
> >probability of a design which is strong against known attacks, but weak
> >against a future attack someone might discover.

This is an extremely radical statement. Do you know of others who argue in the same vein?

Personally, I have often expressed the opinion that the biggest security risk in cipher design is the possible discovery of a catastrophic attack method against cipher designs considered strong today. A catastrophic attack would be an attack that can be used in _practice_ to uncover the secret key based on only a handful of known plaintexts. If this should happen in the future and work against a standard cipher, the repercussions could be worse than the Y2K error.

Now I have argued that a possible defense against unknown attacks are ciphers that have as little internal structure as possible. My reasoning is that a catastrophic attack will probably take advantage of some characteristic or weakness of the cipher's structure. If a cipher has little structure then it will be less likely to have that weakness. Now, what you are saying is I think more radical: you are saying that current cipher design methodology based on analysis against known attacks not only fails to strengthen the new ciphers against unknown attacks but actually makes them weaker.

Super-encipherment, where several distinct ciphers, preferably with distinct design philosophies, are combined in series is another albeit slower defense against unknown attacks. The reasoning is that it is unlikely that an attack would be powerful enough to penetrate all different key-schedule methods and layers of rounds. There is another advantage here: there may exist a "General Cryptanalytic Theory" that can be used to analyze and catastrophically break _any_ cipher whose workload is bellow some limit, i.e. any cipher that is fast enough. A slow and complex "Super-Cipher" would hopefully exceed this limit. I wonder if concurrently to the fast AES, we shouldn't have a standard Superencipherment algorithm scalable in speed. Really important security could then be done at orders of magnitude less speed than the AES, possibly at a few kilobytes per second on a PC.

----------== Posted via Deja News, The Discussion Network ==----------
http://www.dejanews.com/ Search, Read, Discuss, or Start Your Own

---

Subject: Re: Thought question: why do public ciphers use only simple ops like shift and XOR?
Date: Thu, 22 Apr 1999 18:48:07 GMT
From: jsavard@tenMAPSONeerf.edmonton.ab.ca (John Savard)
Message-ID: <371f6e1c.13388611@news.prosurfr.com>
References: <7fn9el$e2v$1@nnrp1.dejanews.com>
Newsgroups: sci.crypt
Lines: 26

dianelos@tecapro.com wrote, in part:
>In article <371a56a8.198396@news.prosurfr.com>,
> jsavard@tenMAPSONeerf.edmonton.ab.ca (John Savard) wrote:

>> >- And I think you can see why this design process actually _increases_ the
>> >probability of a design which is strong against known attacks, but weak
>> >against a future attack someone might discover.

> This is an extremely radical statement. Do you know of others who
> argue in the same vein?

I think it may have seemed more radical than it was. I wasn't saying that modifying a cipher design in response to cryptanalysis makes it weaker, which would be not only very radical, but quite wrong.

What I had meant was that if we consider acceptable a design that was initially weak against known attacks, but which was modified later specifically to resist them, then we are, I think, with that kind of design more likely to fall prey to *new* attacks than we would be with a design that was resistant to the known attacks right from the start.

And I don't think that _that_ is a particularly radical statement, but I could be wrong.

John Savard (teneerf<-)
http://members.xoom.com/quadibloc/index.html

Subject: Re: Thought question: why do public ciphers use only simple ops like shift and XOR?
Date: Thu, 22 Apr 1999 19:51:14 GMT
From: SCOTT19U.ZIP_GUY <dscott@networkusa.net>

Message-ID: <7fnujg$2tj$1@nnrp1.dejanews.com>
References: <7fn9el$e2v$1@nnrp1.dejanews.com>
Newsgroups: sci.crypt
Lines: 65

In article <7fn9el$e2v$1@nnrp1.dejanews.com>,
  dianelos@tecapro.com wrote:
>...
> Personally, I have often expressed the opinion that the biggest
> security risk in cipher design is the possible discovery of a
> catastrophic attack method against cipher designs considered
> strong today. A catastrophic attack would be an attack that can be
> used in _practice_ to uncover the secret key based on only a
> handful of known plaintexts. If this should happen in the future
> and work against a standard cipher, the repercussions could be
> worse than the Y2K error.
>
> Now I have argued that a possible defense against unknown attacks
> are ciphers that have as little internal structure as possible. My
> reasoning is that a catastrophic attack will probably take
> advantage of some characteristic or weakness of the cipher's
> structure. If a cipher has little structure then it will be less
> likely to have that weakness. Now, what you are saying is I think
> more radical: you are saying that current cipher design
> methodology based on analysis against known attacks not only fails
> to strengthen the new ciphers against unknown attacks but actually
> makes them weaker.
>
> Super-encipherment, where several distinct ciphers, preferably
> with distinct design philosophies, are combined in series is
> another albeit slower defense against unknown attacks. The
> reasoning is that it is unlikely that an attack would be powerful
> enough to penetrate all different key-schedule methods and layers
> of rounds. There is another advantage here: there may exist a
> "General Cryptanalytic Theory" that can be used to analyze and
> catastrophically break _any_ cipher whose workload is bellow some
> limit, i.e. any cipher that is fast enough. A slow and complex
> "Super-Cipher" would hopefully exceed this limit. I wonder if
> concurrently to the fast AES, we shouldn't have a standard
> Superencipherment algorithm scalable in speed. Really important
> security could then be done at orders of magnitude less speed than
> the AES, possibly at a few kilobytes per second on a PC.
>

  Your feelings are correct except for one small point. The AES contest
is not about having secure encryption. The NSA would never allow a good
common method to be blessed by the government for general use.
 So that is why you will never see a blessed super cipher method made
of completely different methods. Unless each method added information
at each pass so that they could be broken independitly.

  If you put such a package together be sure to add scott16u or
scott19u in it. In each a have a mode where the file size out
matchs the file size in. This should be a requirement for your
approach to a super-cipher. If the method cahnges the file length
it is to hard to check for NSA approved added info to the methods
so that it could be broken.
  I also think even my hate mongers would agree my methods are very
different than the so called blessed methods.

David A. Scott
--
http://cryptography.org/cgi-bin/crypto.cgi/Misc/scott19u.zip
http://members.xoom.com/ecil/index.htm
NOTE EMAIL address is for SPAMERS
to email me use address on WEB PAGE

----------== Posted via Deja News, The Discussion Network ==----------
http://www.dejanews.com/ Search, Read, Discuss, or Start Your Own

---

Subject: Re: Thought question: why do public ciphers use only simple ops like shift
and XOR?
Date: Sat, 24 Apr 1999 01:36:38 -0600
From: jgfunj@vgrknf.arg (wtshaw)
Message-ID: <jgfunj-2404990136390001@dial-243-115.itexas.net>
References: <7fq06b$t83$1@nnrp1.dejanews.com>
   <jgfunj-2304990140240001@dial-243-089.itexas.net>
   <7fnujg$2tj$1@nnrp1.dejanews.com>
Newsgroups: sci.crypt
Lines: 83

In article <7fq06b$t83$1@nnrp1.dejanews.com>, SCOTT19U.ZIP_GUY
<dscott@networkusa.net> wrote:

> In article <jgfunj-2304990140240001@dial-243-089.itexas.net>,
> jgfunj@vgrknf.arg (wtshaw) wrote:
> > In article <7fnujg$2tj$1@nnrp1.dejanews.com>, SCOTT19U.ZIP_GUY
> > <dscott@networkusa.net> wrote:
> > >
> > > Your feelings are correct except for one small point. The AES contest
> > > is not about having secure encryption. The NSA would never allow a good
> > > common method to be blessed by the government for general use.

> > > So that is why you will never see a blessed super cipher method made
> > > of completely different methods. Unless each method added information
> > > at each pass so that they could be broken independitly.
> > >
> > We can expect that there are tactics and strategies on file for finding a
> > way for the government to get itself out of the corner that it has painted
> > itself into. Whether any of them will work or is going to be acceptable
> > is up for grabs.
> >
> > As seen in many cases before, our government is adept at doing
> > inconsistent things, even, alas, contradicting itself in statements of
> > mission and policy. As in any level of organization, individual to
> > government, the devil is in the details, that is which values are given
> > more weight at the time. In time of war....well, we will see...rather not
> > have to have any war that might be abused for ulterior motives aside from
> > those well recognized.
>
> These inconsistent things are done for a reason. One branch of the
> government can promise you something in exchange for something but
> when you keep your promise then another branch can punish you for the
> very same action. That way those in power can really do what the hell
> they want because there are so many laws rules and organiszations
> they are routinely selectively applied. As an example is the tax
> structure. It cost a lot of money to collect and process the taxs
> any comgress man knows a simple flat tax would save the government
> money and be best for the population as a whole and it could make
> the size of government smaller. But this will never happen it is
> designed to keep citizens in fear instead of helping them. Also
> as a recent stufy showed for a family of 4 with a simple tax set
> up the mahority of CPA's could not do the tax forms correctly.
> The govenment likes this. If you vote wrong or whatevery you can
> be charged with falsefying your income taxes. This tool is to
> powerful for government to throw away. The governments main
> concern is to keep the status quo and to control. Not to help
> people.

I agree with you about most of what you have said above, but it may be
more true of some in government than others. We could add lots of other
examples, at various levels of government. The best thing to be is open
and honest, something government does not particularly like to do as it
limits arbitrary power.
>
> Next silly thing your going to tell me is that if the Pres
> comitted perjury he would be punished in court like any other
> man.

Lots of wrong statements make it into the record by default.
Inconsistencies are overlooked because they are so common, and the
workloads demand getting on to something else. And, perjury is often
expected to have occured at some point; otherwise, anyone protesting
charges were untrue could be charged with perjury if there was a
conviction.

In Clinton's case, due process was carried out, be it a special form.

> I still think the shootings in Colorado are more the result
> of the mixed signals we send our kids. We teach them about
> truth and justice and fair play. But then the first taste of
> justice is a dishonest traffic ticket.

We have many good rules about how school should be. The school
authorities do not follow them too closely all too often. When problems
are not addressed, the odds that trouble will occur just increase. What
should be done is an open question, every contemplated action has its down
side; formal demands for guaranteeing a good education environment are all
too often treated as mere guidelines. It is the same story as it was with
the courts, time constraints are real in the schools; but, that is no
excuse for not doing more to address real problems that affect each
individual, and individuals do reach breaking points.
--
Life's battles do not always go to the stronger of faster man...
But, sooner or later always go to the fellow who thinks he can.

Subject: Re: Thought question: why do public ciphers use only simple ops like shift
and XOR?
Date: Sun, 25 Apr 1999 00:34:37 -0600
From: jgfunj@vgrknf.arg (wtshaw)
Message-ID: <jgfunj-2504990034380001@dial-243-101.itexas.net>
References: <3721B517.ED9E4D4F@null.net>
   <jgfunj-2404990136390001@dial-243-115.itexas.net>
Newsgroups: sci.crypt
Lines: 28

In article <3721B517.ED9E4D4F@null.net>, "Douglas A. Gwyn"
<DAGwyn@null.net> wrote:

> wtshaw wrote:
> > We have many good rules about how school should be. The school
> > authorities do not follow them too closely all too often.
>
> I wouldn't place most of the blame on the educational system, bad
> as it is, when parents allow their kids to stray so far off-track
> as in the Columbine case.

>
> Anyway, this is off-topic. I presume there is some better place
> for such discussions.

Topics sometimes wander...you're right.

Back to topic, simple answer to the question in the title: Monkey see, monkey do.

This is not meant to be a slam, but a clear fact that if you only learn a few selected crypto primatives, those are the ones you are apt to utilize. The same applies to almost any area of endeavor as well. Only when you get beyond following in someelse's wake will you feel the freedom to explore the unknown, which means finding different ways of doing things, perhaps even a few better ones mixed in.
--
Life's battles do not always go to the stronger of faster man...
But, sooner or later always go to the fellow who thinks he can.

Subject: Re: Thought question: why do public ciphers use only simple ops like shift and XOR?
Date: Sun, 25 Apr 1999 20:56:52 GMT
From: bryan.olson@uptronics.com
Message-ID: <7fvvii$pu7$1@nnrp1.dejanews.com>
References: <37232059.4FA1@sundialservices.com>
Newsgroups: sci.crypt
Lines: 14

In article <37232059.4FA1@sundialservices.com>,
  info@sundialservices.com wrote:
> > : olson@umbc.edu (Bryan G. Olson; CMSC (G)) wrote, in part:
> [...]
> > : However, I don't think it's appropriate to automatically conclude that
[...]

I (Bryan) only wrote that in quoting its author, John Savard.

--Bryan

----------== Posted via Deja News, The Discussion Network ==----------
http://www.dejanews.com/ Search, Read, Discuss, or Start Your Own

---

Subject: Re: Thought question: why do public ciphers use only simple ops like shift and XOR?
Date: Mon, 26 Apr 1999 05:38:03 GMT
From: dianelos@tecapro.com
Message-ID: <7g0u3r$j1h$1@nnrp1.dejanews.com>
References: <7fusfv$as8$1@news.umbc.edu>
Newsgroups: sci.crypt
Lines: 96

In article <7fusfv$as8$1@news.umbc.edu>,
  olson@umbc.edu (Bryan G. Olson; CMSC (G)) wrote:
> ...
> There is a situation worse than having all one's eggs in one basket.
> The problem with one basket is that there exists a potential failure
> that would be catastrophic. What's worse is a system in which any
> one of many possible failures would be catastrophic. If one accepts
> that in realistic applications of cryptography the same intelligence
> is available from many messages, then choosing from a thousand
> ciphers for each message moves us from one potential catastrophic
> failure to many potential catastrophic failures.

I think you assume that the attacker will know which cipher has been used. In fact, a good variable cipher protocol would hide the information about which cipher or cipher combination has been used.

Let us design two possible future worlds and then pick the one that is more secure:

In the first the AES is used for almost all encryption.

In the second world we define a set of several interesting ciphers, preferably ciphers that are different in some fundamental ways. We put in there the AES, some more ciphers that have been extensively analyzed, some ciphers that follow different design methodologies (for example variable ciphers such as Frog, ciphers designed specifically for making analysis very difficult, ciphers using novel primitives or structures, etc.). Now add to all encrypted data or make implicit in all security applications the following information: the subset of the ciphers that must be used and at which "depth", i.e. how many ciphers out of this subset are cascaded in series. Finally extend the secret key with a sufficient number of bits that define the sequence of the ciphers. (I don't want to discuss here how the individual ciphers' keys are defined - I know it is not optimal but as a first approximation let us suppose all individual keys are identical.) Now observe that if you want to use the AES, you just define a subset that includes only the AES and a depth of one. But you can also include the entire set and a depth of one hundred.

In fact, the original set of ciphers need not be fixed. Allow anybody to add his or her code to the lot in a public Internet server in an environment where everybody can "Amazon-like" comment on all present ciphers, where the experts' opinion is expressively stated and where statistics are held about which products include which ciphers in their "standard" set of ciphers. If my email program receives a message that uses a subset with a cipher not present in my computer, then it will download the authenticated Java code from the public server before decrypting.

So which world do you think is more secure: the AES-centric one, or the "organized chaos" one? It seems to me the latter, because the attacker will have a more complex task and less information to work with.

Now, even if we agree that the organized chaos world is more secure we still have to discuss costs. Now observe that people can always use only optimized AES code in their applications (in fact, the public server could also include optimized code for several cipher/platform combinations). In all cases, royalty free Java code for the cascaded ciphers would not really increase the cost of an application. Many secure applications could even be cipher-neutral. For example, a paranoid organization could use a standard email program and define a large subset of ciphers at a great depth without really paying anything more. So I think the increase of costs would really be marginal and would correspond largely to the definition of a standard protocol for cascading ciphers as well as the operation of the "cipher server". In fact in some cases there may be some cost advantages. For example, suppose RC6 is chosen as the AES but this cipher can not be as easily ported to smartcards as RC6a. In the organized chaos world a smartcard manufacturer could use the cheaper RC6a even in applications where this smartcard will communicate with PCs all over the world.

One can ask if all this is really necessary. After all most experts think that it is extremely unlikely that the AES will suffer a catastrophic failure in the next 50 years or so. Even so, it is deeply troubling to think that we are moving towards an information society and that much of its security will not be based on theoretical proof or, at least, experimental test but on personal opinion of a few dozen people, even if they are excellent and well meaning professionals.

It is still possible that somebody will publish a provable secure cipher that is practical to implement. Meanwhile a variable cipher protocol similar to the one described above would fulfil almost everybody's requirements for symmetric encryption. This would leave many other problems to worry about such as key management and public key systems, Trojan horses, the appropriate use of encryption technology, etc.

----------== Posted via Deja News, The Discussion Network ==----------
http://www.dejanews.com/ Search, Read, Discuss, or Start Your Own

---

Subject: Re: Thought question: why do public ciphers use only simple ops like shift and XOR?
Date: Mon, 26 Apr 1999 15:58:37 -0600
From: jcoffin@taeus.com (Jerry Coffin)
Message-ID: <MPG.118e89fdf2aca1bd989a25@news.rmi.net>
References: <7g0u3r$j1h$1@nnrp1.dejanews.com>
Newsgroups: sci.crypt
Lines: 132

In article <7g0u3r$j1h$1@nnrp1.dejanews.com>, dianelos@tecapro.com says...

[...]

> Let us design two possible future worlds and then pick the one
> that is more secure:
>
> In the first the AES is used for almost all encryption.
>
> In the second world we define a set of several interesting
> ciphers, preferably ciphers that are different in some fundamental
> ways. We put in there the AES, some more ciphers that have been
> extensively analyzed, some ciphers that follow different design
> methodologies (for example variable ciphers such as Frog, ciphers
> designed specifically for making analysis very difficult, ciphers
> using novel primitives or structures, etc.). Now add to all
> encrypted data or make implicit in all security applications the
> following information: the subset of the ciphers that must be used
> and at which "depth", i.e. how many ciphers out of this subset are
> cascaded in series. Finally extend the secret key with a
> sufficient number of bits that define the sequence of the ciphers.
> (I don't want to discuss here how the individual ciphers' keys are
> defined - I know it is not optimal but as a first approximation
> let us suppose all individual keys are identical.) Now observe
> that if you want to use the AES, you just define a subset that
> includes only the AES and a depth of one. But you can also include

> the entire set and a depth of one hundred.

The first is more secure, or at least more dependably secure. The problem is, when you combine two algorithms, you're basically designing a new cypher. If you're lucky, it'll combine the strengths of both the base cyphers, while negating some of the weaknesses of each.

Unfortunately, in cryptology luck tends to be of the bad kind -- you might combine two cyphers that end up negating each other's good points, and nearly eliminating each other strengths.

Ultimately, when somebody designs something like DES, IDEA or Blowfish, they're combining a number of more primitive operations into a single, complete cypher.

In your scenario, essentially the same thing is happening, EXCEPT that instead of an expert in cryptography studying the individual primitives in detail to ensure that they produce a good output, in the typical scenario somebody who knows nothing about cryptography is going to combine things with little or no chance to study them at all. The result may be quite secure, but it may also be EXTREMELY insecure. Without doing a fairly intensive study of the exact combination used, it's nearly impossible to say which.

> In fact, the original set of ciphers need not be fixed. Allow
> anybody to add his or her code to the lot in a public Internet
> server in an environment where everybody can "Amazon-like" comment
> on all present ciphers, where the experts' opinion is expressively
> stated and where statistics are held about which products include
> which ciphers in their "standard" set of ciphers.

This gets worse and worse. Rather than having a small set of primitives that you _might_ be able to study in detail, you're now combining an unknown number of primitives in completely unknown ways. There's simply no way that anybody can keep up with all the possible combinations and figure out which of them produce dangerously poorly encrypted output.

> So which world do you think is more secure: the AES-centric one,
> or the "organized chaos" one? It seems to me the latter, because
> the attacker will have a more complex task and less information to
> work with.

It seems to me the former. Ultimately, you're designing a single cypher that'll be used to do the job. You're simply taking the design of the cypher away from people who study for years about how to do it as well as possible, and instead putting it in the hands of (mostly) people who haven't a clue of how to design a cypher.

> One can ask if all this is really necessary.

One _should_ start by asking whether it's really useful. Since the answer is "only rarely, if ever", it's pointless to deal with costs, necessity, etc.

> It is still possible that somebody will publish a provable secure
> cipher that is practical to implement. Meanwhile a variable cipher
> protocol similar to the one described above would fulfil almost
> everybody's requirements for symmetric encryption. This would leave
> many other problems to worry about such as key management and
> public key systems, Trojan horses, the appropriate use of encryption
> technology, etc.

First of all, to a limited degree, variable protocols such as you describe are already available -- for example, different versions of PGP support triple-DES either or IDEA for the encryption, and RSA or Diffie-Hellman for key exchange. Secure email-protocols support relatively open-ended descriptions of the encryption used in a particular message, though (thankfully) only a few forms of encryption are presently supported.

Second, the variable protocol you propose would be likely to fulfill people's needs under only two possible sets of circumstances: either 1) everybody becomes an expert in designing encryption before they use it, or 2) they really don't need much security in the first place.

DES, AES, etc., are all about one basic idea: since most people neither know, nor want to know how to design secure encryption, the people who do know and care design something that nearly anybody can use, and derive real usefulness from it. If you take a number of components of poorly understood design, and leave it to a total amateur to pick a combination that'll work well, chances are all too high that the final result will be catastrophically awful.

In short: if you intend to combine cyphers and get a secure result, you need to put in quite a bit of study to know how they may interact. Just for example, most people take for granted that triple-DES is more secure than DES, simply because you have three times as large of a key. In fact, in the case of DES, it IS true, because it's been proven that DES does not form a group.

By contrast, assume somebody has the mistaken assumption that it all really comes down to key-size (quite a common misconception). He notices that the "XOR stream encryption" module will allow him to

enter MUCH larger keys than any of the others, so he decides to do a "triple XOR stream encryption" with three different 40-byte (320-bit) keys.

Now, it happens that a simple XOR stream encryption DOES form a group, so first of all, doing it three times with three different keys hasn't really accomplished a thing -- you've still basically got only a 40-byte key. As I'm sure you're well aware, a simple XOR encryption with a 40-byte key is _pathetically_ easy to break, even for a rank amateur at cryptanalysis.

Subject: Re: Thought question: why do public ciphers use only simple ops like shift and XOR?
Date: Tue, 27 Apr 1999 09:25:50 -0400
From: "Trevor Jackson, III" <fullmoon@aspi.net>
Message-ID: <3725BADE.EF3B0685@aspi.net>
References: <MPG.118e89fdf2aca1bd989a25@news.rmi.net>
Newsgroups: sci.crypt
Lines: 143

Jerry Coffin wrote:
>
> In article <7g0u3r$j1h$1@nnrp1.dejanews.com>, dianelos@tecapro.com
> says...
>
> [...]
>
> > Let us design two possible future worlds and then pick the one
> > that is more secure:
> >
> > In the first the AES is used for almost all encryption.
> >
> > In the second world we define a set of several interesting
> > ciphers, preferably ciphers that are different in some fundamental
> > ways. We put in there the AES, some more ciphers that have been
> > extensively analyzed, some ciphers that follow different design
> > methodologies (for example variable ciphers such as Frog, ciphers
> > designed specifically for making analysis very difficult, ciphers
> > using novel primitives or structures, etc.). Now add to all
> > encrypted data or make implicit in all security applications the
> > following information: the subset of the ciphers that must be used
> > and at which "depth", i.e. how many ciphers out of this subset are
> > cascaded in series. Finally extend the secret key with a
> > sufficient number of bits that define the sequence of the ciphers.
> > (I don't want to discuss here how the individual ciphers' keys are
> > defined - I know it is not optimal but as a first approximation
> > let us suppose all individual keys are identical.) Now observe
> > that if you want to use the AES, you just define a subset that
> > includes only the AES and a depth of one. But you can also include
> > the entire set and a depth of one hundred.
>
> The first is more secure, or at least more dependably secure. The
> problem is, when you combine two algorithms, you're basically
> designing a new cypher. If you're lucky, it'll combine the strengths
> of both the base cyphers, while negating some of the weaknesses of
> each.
>
> Unfortunately, in cryptology luck tends to be of the bad kind -- you
> might combine two cyphers that end up negating each other's good
> points, and nearly eliminating each other strengths.
>
> Ultimately, when somebody designs something like DES, IDEA or
> Blowfish, they're combining a number of more primitive operations into
> a single, complete cypher.
>
> In your scenario, essentially the same thing is happening, EXCEPT that
> instead of an expert in cryptography studying the individual
> primitives in detail to ensure that they produce a good output, in the
> typical scenario somebody who knows nothing about cryptography is
> going to combine things with little or no chance to study them at all.
> The result may be quite secure, but it may also be EXTREMELY insecure.
> Without doing a fairly intensive study of the exact combination used,
> it's nearly impossible to say which.
>
> > In fact, the original set of ciphers need not be fixed. Allow
> > anybody to add his or her code to the lot in a public Internet
> > server in an environment where everybody can "Amazon-like" comment
> > on all present ciphers, where the experts' opinion is expressively
> > stated and where statistics are held about which products include
> > which ciphers in their "standard" set of ciphers.
>
> This gets worse and worse. Rather than having a small set of
> primitives that you _might_ be able to study in detail, you're now
> combining an unknown number of primitives in completely unknown ways.
> There's simply no way that anybody can keep up with all the possible
> combinations and figure out which of them produce dangerously poorly
> encrypted output.
>
> > So which world do you think is more secure: the AES-centric one,
> > or the "organized chaos" one? It seems to me the latter, because
> > the attacker will have a more complex task and less information to
> > work with.
>

> It seems to me the former. Ultimately, you're designing a single
> cypher that'll be used to do the job. You're simply taking the design
> of the cypher away from people who study for years about how to do it
> as well as possible, and instead putting it in the hands of (mostly)
> people who haven't a clue of how to design a cypher.
>
> >   One can ask if all this is really necessary.
>
> One _should_ start by asking whether it's really useful. Since the
> answer is "only rarely, if ever", it's pointless to deal with costs,
> necessity, etc.
>
> >   It is still possible that somebody will publish a provable secure
> >   cipher that is practical to implement. Meanwhile a variable cipher
> >   protocol similar to the one described above would fulfil almost
> >   everybody's requirements for symmetric encryption. This would leave
> >   many other problems to worry about such as key management and
> >   public key systems, Trojan horses, the appropriate use of encryption
> >   technology, etc.
>
> First of all, to a limited degree, variable protocols such as you
> describe are already available -- for example, different versions of
> PGP support triple-DES either or IDEA for the encryption, and RSA or
> Diffie-Hellman for key exchange. Secure email-protocols support
> relatively open-ended descriptions of the encryption used in a
> particular message, though (thankfully) only a few forms of encryption
> are presently supported.
>
> Second, the variable protocol you propose would be likely to fulfill
> people's needs under only two possible sets of circumstances: either
> 1) everybody becomes an expert in designing encryption before they use
> it, or 2) they really don't need much security in the first place.
>
> DES, AES, etc., are all about one basic idea: since most people
> neither know, nor want to know how to design secure encryption, the
> people who do know and care design something that nearly anybody can
> use, and derive real usefulness from it. If you take a number of
> components of poorly understood design, and leave it to a total
> amateur to pick a combination that'll work well, chances are all too
> high that the final result will be catastrophically awful.
>
> In short: if you intend to combine cyphers and get a secure result,
> you need to put in quite a bit of study to know how they may interact.
> Just for example, most people take for granted that triple-DES is more
> secure than DES, simply because you have three times as large of a
> key. In fact, in the case of DES, it IS true, because it's been
> proven that DES does not form a group.
>
> By contrast, assume somebody has the mistaken assumption that it all
> really comes down to key-size (quite a common misconception). He
> notices that the "XOR stream encryption" module will allow him to
> enter MUCH larger keys than any of the others, so he decides to do a
> "triple XOR stream encryption" with three different 40-byte (320-bit)
> keys.
>
> Now, it happens that a simple XOR stream encryption DOES form a group,
> so first of all, doing it three times with three different keys hasn't
> really accomplished a thing -- you've still basically got only a 40-
> byte key. As I'm sure you're well aware, a simple XOR encryption with
> a 40-byte key is _pathetically_ easy to break, even for a rank amateur
> at cryptanalysis.

You've made some strong claims here, and demolished a trivial example.
Can you show a real example? Are there _any_ known weaknesses in
combining any pair of the following: Blowfish, IDEA, 3DES? An easier
question would be to ask whther there are any weaknesses known in
combining one of the previously mentioned list with any other cipher.

Are there real facts behind your claims or are you expressing a
subjective judgement?

Subject: Re: Thought question: why do public ciphers use only simple ops like shift
and XOR?
Date: Tue, 27 Apr 1999 22:15:30 -0600
From: jcoffin@taeus.com (Jerry Coffin)
Message-ID: <MPG.11902efa82856502989a2e@news.rmi.net>
References: <3725BADE.EF3B0685@aspi.net>
  <MPG.118e89fdf2aca1bd989a25@news.rmi.net>
Newsgroups: sci.crypt
Lines: 63

In article <3725BADE.EF3B0685@aspi.net>, fullmoon@aspi.net says...

[...]

> You've made some strong claims here, and demolished a trivial example.
> Can you show a real example? Are there _any_ known weaknesses in
> combining any pair of the following: Blowfish, IDEA, 3DES?

> An easier
> question would be to ask whther there are any weaknesses known in
> combining one of the previously mentioned list with any other cipher.

> Are there real facts behind your claims or are you expressing a
> subjective judgement?

Of course there are real facts. I've demonstrated how one trivial example shows up poorly -- I'd thought that would show the fundamental problem, but apparently it didn't, so I'll try to point out the most fundamental problem more explicitly.

The most fundamental problem with the idea as-presented is that all the forms of encryption use the same key. That means that if any _one_ of the forms of encryption used is broken, the enemy can recover the key and decrypt the message.

Just for the sake of demonstration, assume that you've got a message that you want to ensure against being decrypted within 10 years. For the sake of argument, let's say you use Blowfish, IDEA and triple-DES. Again, purely for the sake of having some numbers, let's assign a 20% chance to each of these three forms of encryption being broken within the ten year time-frame.

In this case, encrypting with one of the three means you have a 20% chance of your message being decrypted. Encrypting with all three means you have a 60% chance of the message being decrypted.

IOW, by using the same key for all forms of encryption, the best you can hope for is to get the overall strength of the single _weakest_ form of encryption used. If any _one_ form of encryption is broken, the message can be decrypted.

To be at all useful, you need to start by using an independent key for each form of encryption. This means breaking one has no effect on the others.

Then you hope that applying the forms of encryption together doesn't lead to some previously unknown/unexpected weakness. Without fairly extensive study of a specific combination, it's impossible to make specific comments, but I'll point out a basic concept to consider: the forms of encryption we're talking about are all symmetric. If you apply essentially the same form of encryption twice, the second can end up decrypting the first. When you combine the two, you're basically hoping for a sum of their individual strengths, but if one ends up partially decrypting the other, you can end up with the difference instead of the sum, with results you'd rather not know about.

Of course, using independent keys for the different forms of encryption CAN help a great deal in this regard as well -- as long as the two forms of encryption don't form a co-group (so to speak) you can end up with a stronger result even if both using the same key would partly or completely cancel each other out. OTOH, if the two form a co-group, using a separate key for each might not help at all.

Subject: Re: Thought question: why do public ciphers use only simple ops like shift and XOR?
Date: Wed, 28 Apr 1999 05:18:00 GMT
From: dianelos@tecapro.com
Message-ID: <7g65m6$9uo$1@nnrp1.dejanews.com>
References: <MPG.118e89fdf2aca1bd989a25@news.rmi.net>
<7g0u3r$j1h$1@nnrp1.dejanews.com>
Newsgroups: sci.crypt
Lines: 103

In article <MPG.118e89fdf2aca1bd989a25@news.rmi.net>,
jcoffin@taeus.com (Jerry Coffin) wrote:
> In article <7g0u3r$j1h$1@nnrp1.dejanews.com>, dianelos@tecapro.com
> says...
>
> [...]
>
> > Let us design two possible future worlds and then pick the one
> > that is more secure:
> >
> > In the first the AES is used for almost all encryption.
> >
> > In the second world we define a set of several interesting
> > ciphers, preferably ciphers that are different in some fundamental
> > ways. We put in there the AES, some more ciphers that have been
> > extensively analyzed, some ciphers that follow different design
> > methodologies (for example variable ciphers such as Frog, ciphers
> > designed specifically for making analysis very difficult, ciphers
> > using novel primitives or structures, etc.). Now add to all
> > encrypted data or make implicit in all security applications the
> > following information: the subset of the ciphers that must be used
> > and at which "depth", i.e. how many ciphers out of this subset are
> > cascaded in series. Finally extend the secret key with a
> > sufficient number of bits that define the sequence of the ciphers.
> > (I don't want to discuss here how the individual ciphers' keys are
> > defined - I know it is not optimal but as a first approximation
> > let us suppose all individual keys are identical.) Now observe
> > that if you want to use the AES, you just define a subset that
> > includes only the AES and a depth of one. But you can also include
> > the entire set and a depth of one hundred.

```
>
> The first is more secure, or at least more dependably secure.  The
> problem is, when you combine two algorithms, you're basically
> designing a new cypher.  If you're lucky, it'll combine the strengths
> of both the base cyphers, while negating some of the weaknesses of
> each.
>
> Unfortunately, in cryptology luck tends to be of the bad kind -- you
> might combine two cyphers that end up negating each other's good
> points, and nearly eliminating each other strengths.

    It is possible that the combination of several independently
    designed ciphers is weaker than each one of them, but it is also
    highly unlikely. It is also possible to break a 3DES encryption by
    guessing the correct key on the first try, but again this is very
    unlikely. Unless someone finds a proof for a cipher's security the
    best we can do is estimate probabilities. Cascading ciphers does
    increase security in this sense. Schneier discusses this method in
    chapter 15.7 of the second edition of Applied Cryptography. There
    are also two papers by Maurer discussing the combination of
    block ciphers and of stream ciphers.

> [...]

Dianelos wrote:
> >     In fact, the original set of ciphers need not be fixed. Allow
> >     anybody to add his or her code to the lot in a public Internet
> >     server in an environment where everybody can "Amazon-like" comment
> >     on all present ciphers, where the experts' opinion is expressively
> >     stated and where statistics are held about which products include
> >     which ciphers in their "standard" set of ciphers.
>
> This gets worse and worse.  Rather than having a small set of
> primitives that you _might_ be able to study in detail, you're now
> combining an unknown number of primitives in completely unknown ways.
> There's simply no way that anybody can keep up with all the possible
> combinations and figure out which of them produce dangerously poorly
> encrypted output.

    Exactly right. You see the point is that if you use a set of eight
    ciphers, you cascade 50 executions of them and you keep the
    sequence secret you end up with about 2^150 _different_ ciphers.
    Also observe that the attacker will not know which of these ciphers
    is used. No possible adversary will be able to analyze any
    significant subset of this monstrous number of ciphers in order to
    mount an attack if, incredibly, a weak combination was found and
    also if, incredibly, this precise combination was used by
    somebody.

    BTW I allow any cipher to be included in the Internet server as a
    matter of practicality. After all, all known ciphers are published
    somewhere and by having them all in one place my email program
    will be able to decrypt anything. Surely the experts would
    recommend which sub-sets of these ciphers should be used in
    practice.

> [...]

Dianelos wrote:
> >     One can ask if all this is really necessary.
>
> One _should_ start by asking whether it's really useful.  Since the
> answer is "only rarely, if ever", it's pointless to deal with costs,
> necessity, etc.

    On the contrary, such a scheme would be useful in almost all cases
    where speed in not an issue. There are many important cases where
    this is the case, such as email or financial transactions.

[...]

-----------== Posted via Deja News, The Discussion Network ==----------
http://www.dejanews.com/       Search, Read, Discuss, or Start Your Own
```

```
Subject: Re: Thought question:  why do public ciphers use only simple ops like shift
and XOR?
Date: Sat, 17 Apr 1999 14:25:52 GMT
From: "Douglas A. Gwyn" &LT;DAGwyn@null.net>
Message-ID: &LT;3718A7C9.12B5EEF@null.net>
References: &LT;3718324d.13916819@news.io.com>
Newsgroups: sci.crypt
Lines: 26

Terry Ritter wrote:
> Currently, cryptanalysis seems to be some sort of "one shot" contest
> against the cryptographer, as opposed to an interactive joint process
> to attain a better cipher.

That is a management problem at the developing organization.

> Even so, we still don't know that their guys aren't better, or even
> just luckier.  I think it sometimes just takes a particular point of
> view to enable an alternative -- possibly much easier -- attack.  And
```

> it may not be the smartest guy who has that new point of view.

Yes, but if your tiger team is really good and experienced
(another management issue), you can attain a certain degree
of confidence based on their positive evaluation (assuming
they didn't have uneasy feelings about undiscovered weaknesses).

> I guess I would suggest that if the goal was to get the best
> cipher, we would see a post-analysis re-design phase intended
> to fix known problems, with the final comparison being made
> between full-strength designs.

I guess you're talking about AES. If time constraints allow,
that would be one reasonable part of the evaluation procedure,
but you still have to drawn the line somewhere and pick the
best-to-date.

Subject: Re: Thought question: why do public ciphers use only simple ops like shift
and XOR?
Date: Sun, 18 Apr 1999 16:49:26 -0400
From: Uri Blumenthal <uri@watson.ibm.com>
Message-ID: <371A4556.F9D193FB@watson.ibm.com>
References: <3718324d.13916819@news.io.com>
Newsgroups: sci.crypt
Lines: 7

At least one publicly known cipher (GOST) uses ADD (a more
complex operation)...
--
Regards,
Uri
-=-=-==-=-=-
<Disclaimer>

Subject: Re: Thought question: why do public ciphers use only simple ops like shift
and XOR?
Date: Sat, 17 Apr 1999 14:37:37 GMT
From: "Douglas A. Gwyn" <DAGwyn@null.net>
Message-ID: <3718AA8A.B2B460DE@null.net>
References: <7fa0n5$v4m$1@nnrp1.dejanews.com>
Newsgroups: sci.crypt
Lines: 22

dscott@networkusa.net wrote:
> ... Most people complain that my S-tables are too large. Since the
> size of s-tables in scott18u.zip is effectively larger than a one
> million byte key.

I think the question in their minds is, whether such a large S-table
is *essential* to attain the desired level of security, or whether
other approaches might be comparably secure.

> My feeling is that the method that computers use should involve much
> more operations than what the public is use to. My code treats the
> whole file as a block. Which is something else the current blessed
> methods do not do.

The immediate thought one has is that treating the whole message
would be a problem in many applications where data is generated
in a stream, for example a TELNET session. However, a whole-file
method can be applied to small blocks simply by treating each
block in the sequence as a separate file. That would lose any
benefit from the large-file property, however. Have you any
estimate of the security of your method when used with small
"files" (say, 512 bytes or less)?

Subject: Question on confidence derived from cryptanalysis.
Date: Sat, 17 Apr 1999 15:00:25 -0400
From: Geoff Thorpe <geoff@raas.co.nz>
Message-ID: <3718DA49.90CA4FAD@raas.co.nz>
References: <37181079.5255438@news.io.com>
Newsgroups: sci.crypt
Lines: 219

Hi there,

I have been following this thread with interest, albeit silently for a
while, and for the most part I have enjoyed the intellectual
horn-locking, especially Terry's unconventional but often insightful
contributions. However, good ideas can quickly get buried in slop, or
just unmasked as reactionary drivel as they seem (IMHO) to in Mr
Ritter's post below ...

Terry Ritter wrote:
> Sure they are. As far as I know, Schneier's point has always been
> that cryptanalysis is the way we know a cipher's strength. I'm sure
> he would agree that this is not proof, but I do not agree that it says
> anything at all. The implication that cryptanalysis would like to
> promote is indeed that of tested strength.

You're contributions in this thread seem to have an emerging theme ... that continued testing of a cipher by cryptanalysists (presumably using the "current most widely recognised techniques") does not guarantee some absolute/quantitative strength of the cipher against any attack (which I'm sure we would ALL, including Mr Schneier, agree with). However you also seem to suggest that it gives us no indication of tested strength at all. And here I disagree with you.

You want to sound a cautionary note that we all risk being naive and over-confident in our "cryptanalytic testing" of ciphers - excellent point and it is well taken. However, please do not go so far as to be similarly naive yourself, and to play things out to an theoretical abyss and expect us to follow you there.

History does in fact support the claim that bashing away at problems with the best techniques you can come up with at the time, for a period of time, DOES give some degree of confidence in "strength" that failing to do so does. Here strength is a practical measure, not a theoretical one.

Now no rational person is going to tell you that RSA simply will never be attacked at a much better complexity than the best current factoring techniques. Similarly, no rational person should assure you that attacking DES or triple DES will never improve much beyond brute-force key-searches. However, I will humbly suggest to you we ARE a lot safer against those possibilities than similar risks with newer and less studied techniques - and that history and common sense DO give us the right to those basic assumptions contrary to the gloomy and highly unhelpful view you hold.

A quick glance at any of the big mathematical problems in history, particularly the ones that are simply stated (ie the difficulty is not composed even partially out of obscurity - it looks more like a brick wall than a maze) almost always are either not solved even today, or were solved using techniques much more sophisticated than those available to those who posed the original question and first tried to solve it. Indeed the classical problems have typically given rise to entire branches of mathematics that grew out of a pursuit of that problem.

Fermat's Theorem is the obvious example but there are others too. Someone more up to date with things could clarify, but I think they were trying to refine Andrew Wiles' proof a little to slice a couple of hundred pages off it ... it simply was not solved using a ruler and compass and the odd quadratic here and there. And yes, as I'm sure you're thinking, it IS possible it can be solved with a ruler and compass and the occasional discriminant. But most people will be happy to accept that that is a lot LESS likely to happen than if I just pose a new simply stated differential equation and state it can't be solved in simple terms only to have someone prove me wrong.

Techniques, understanding, and formalised mathematical frameworks evolve - occasionally someone does throw something new and useful in and things accelerate for a while, but sudden breaks solving historical problems with simple techniques are VERY much the exception not the rule.

Let me ask the following - do you disagree with the following statement; "History has demonstrated time and time again, that the longer a problem resists the attack of academics, hobbyists, and mechanics - the probability the problem can be broken using simple techniques that were available at the time the problem was posed (or even comprehensible to the people of that time) decreases."

Occasionally someone invents a wheel, but divine beams of light are a lot less common than simple grunt-work and craftsmanship. This is also true of "our opponents" as you have a tendency to call them.

> >Not at least trying cryptanalysis on a cipher is stupid which
> >I'm sure you agree with.
>
> I do. But there is no one cryptanalysis. Indeed, there is no end to
> it. But we do have to make an end before we can field anything. This
> in itself tells us that cryptanalysis as certification is necessarily
> incomplete.

It is all probabilities and risk management. Mr Schneier will hopefully agree with me on that and I hope you do too (I hope anyone contributing to the crypto-frameworks I will have to use day-to-day agree with that also).

Would you have us believe that all things that are not absolute are necessarily equal? God, this sounds like a debate on socialism all of a sudden - my humblest apologies [;-)

> Our main problem is that cryptanalysis does NOT say that there is no
> simpler attack. It does NOT say that a well-examined cipher is secure
> from your kid sister. Oh, many people will offer their opinion, but
> you won't see many such a claims in scientific papers, because there
> we expect actual facts, as opposed to wishes, hopes, and dreams.

But those claims say as much as; "we've hopefully done the best we can with the best techniques we have and the best people we can find, and this one seemed to resist our best attacks the best so we can only give you the best assurances we can that the best chance you have is to use this one".

If you cannot interpret cryptanalytic conclusions in that fashion then you seem to miss their point. I agree with Mr Schneier ... it is a race - our opponents (again using your phrase) get to see the best cryptanalytic techniques we have, and sometimes we get a peek (or a leak) at theirs ... we just do the best we can with what we've got - and history has shown that if we keep that up for a while, the chances of an about-turn due to some radical improvement in the theory decreases steadily.

> Cryptanalysis does NOT give us an indication of how much effort our
> Opponent will have to spend to break the cipher. Yet that is exactly
> what the cryptanalytic process would like us to believe: That is why

I disagree - your point of view has some merit but is no more valid than the polar opposite statement. If people devote their lives to keeping up to date with the literature and do their best to innovate and develop in full public-view, and their best attempts to break things fail for a period of time (and I'm talking about the crypto community as a whole here) then we CAN infer that that process represents a steadily increasing probability that it's not going to fall over tomorrow in some dramatic fashion. I do not mean that evolving cryptanalysis work provides increasing confidence in brand-new ciphers and what-not, rather that as one cipher builds up a catalogue of evolving cryptanalysis work against it that we DO have a decreasing probability that THAT cipher will fall over in show-stopper fashion.

> we have the process of: 1) design a cipher, and 2) certify the
> cipher by cryptanalysis. As I see it, the real opportunity for
> cryptanalysis is as part of a dynamic and interactive cipher design
> process, as opposed to final certification.

And it currently isn't? What exactly does the open publication of research, countless conferences, news-groups, mail-lists, web-sites, open-source projects, etc amount to other than a dynamic and interactive process? Also, thousands of hobbyists and professionals all doing their damndest to break each others ciphers gives me personally some confidence in the value of "standing the test of time".

> Thanks. I suggest you learn it by heart if you intend to depend upon
> cryptography.

I suggest that you get a little more realistic. What do you have more confidence in, "NT.DLL" or an established release version of the linux kernel? Or IIS versus Apache? (again, speaking about versions which aren't acknowledged by the authors as being "beta"). And no, that question is not rhetorical, I'm actually interested to hear your response.

As for your continued suggestion that confidence in (relative) conclusions reached by noted cryptanalysts is overrated and work by lesser mortals unfairly disregarded. In reality I think you are wrong.
(a) If a lesser mortal finds an improvement in cracking DES keys, they need only publish it to sci.crypt with the header "I think I can hack DES keys a bit faster ...CRYPHTML.HTM" and they will get all the attention to their claims they desire, and if they have the facts to back it up they needn't worry about anonymity. (b) If someone with a track-record proposes a new cipher (or in my metaphor, an alteration to kernel.c in Linux) and someone unknown does the same, it is natural, right, and fair for me to regard the latter with more scepticism and the former with a little more of an open mind.

Perhaps this Darwinist philosophy is not to your liking but I'm afraid it fits the model. If I have a studied knowledge of shooting, am good at it myself, stay abreast of the most modern trends, and am widely respected as an expert in the field - then I am probably as good a person as any to suggest methods for staying out of the firing line.

> This is my bit for public education.

And it has been useful to provide for thoughtful debate - but I think you overreach to absolute conclusions to counter opposing conclusions that I don't think anybody is actually making.

> I have no modern products. I do offer cryptographic consulting time,
> and then I call it as I see it. I also own patented cryptographic
> technology which could be useful in a wide range of ciphers.

Great - perhaps if you would benefit us all (if that is your aim) by describing
(a) how you made design decisions for your cryptographic technology (particularly with relationship to your awareness of classical and modern loopholes and weaknesses you were trying to avoid).
(b) what kind of analysis has been (or could be) done on the/those technology(ies).
(c) how you would convince anybody that your ideas merit some degree of trust/faith/use/investment.

Do you expect us to assume that even though the winning AES candidate will have been subjected to very deep analysis by vary many parties of very different angles of vested interest/disinterest, because it COULD be broken tomorrow it is has no more measurable "strength" than a boutique new idea which has not been widely distributed and tested? The fact two things are neither black or white does not imply they are the same shade of grey.

> I see no problem with someone promoting what they think is an advance
> in the field, even if they will benefit. But when reasoning errors
> are promoted which just happen to benefit one's business -- in fact, a
> whole sub-industry -- some skepticism seems appropriate. Just once I
> would like to see delusions promoted which produce *less* business.

You call them "delusions", I call them "reasoned and qualified critiques
open to public dissemination and review" - let's call the whole thing
off. (as the song goes).

Regards,
Geoff

Subject: Re: Question on confidence derived from cryptanalysis.
Date: Sat, 17 Apr 1999 19:50:05 GMT
From: ritter@io.com (Terry Ritter)
Message-ID: <3718e5e9.9093614@news.io.com>
References: <3718DA49.90CA4FAD@raas.co.nz>
Newsgroups: sci.crypt
Lines: 463

On Sat, 17 Apr 1999 15:00:25 -0400, in <3718DA49.90CA4FAD@raas.co.nz>,
in sci.crypt Geoff Thorpe <geoff@raas.co.nz> wrote:

>Hi there,
>
>I have been following this thread with interest, albeit silently for a
>while, and for the most part I have enjoyed the intellectual
>horn-locking, especially Terry's unconventional but often insightful
>contributions. However, good ideas can quickly get buried in slop, or
>just unmasked as reactionary drivel as they seem (IMHO) to in Mr
>Ritter's post below ...
>
>Terry Ritter wrote:
>> Sure they are. As far as I know, Schneier's point has always been
>> that cryptanalysis is the way we know a cipher's strength. I'm sure
>> he would agree that this is not proof, but I do not agree that it says
>> anything at all. The implication that cryptanalysis would like to
>> promote is indeed that of tested strength.
>
>You're contributions in this thread seem to have an emerging theme ...
>that continued testing of a cipher by cryptanalysists (presumably using
>the "current most widely recognised techniques") does not guarantee some
>absolute/quantitative strength of the cipher against any attack (which
>I'm sure we would ALL, including Mr Schneier, agree with). However you
>also seem to suggest that it gives us no indication of tested strength
>at all. And here I disagree with you.

So here we are in disagreement.

>You want to sound a cautionary note that we all risk being naive and
>over-confident in our "cryptanalytic testing" of ciphers - excellent
>point and it is well taken.

No, the point is NOT well-taken. It is ignored and brushed off as
trivial and known. Then everyone sticks their head in the sand again
until I bring it up again. This has happened for years.

>However, please do not go so far as to be
>similarly naive yourself, and to play things out to an theoretical abyss
>and expect us to follow you there.

The abyss is there. By not following, you are in it.

>History does in fact support the claim that bashing away at problems
>with the best techniques you can come up with at the time, for a period
>of time, DOES give some degree of confidence in "strength" that failing
>to do so does. Here strength is a practical measure, not a theoretical
>one.

But the only thing being "measured" here is the open, academic
analysis. The *real* experts do not play this way. We thus have no
way to understand their capabilities. The strength value measured on
academics cannot apply to the real problem.

>Now no rational person is going to tell you that RSA simply will never
>be attacked at a much better complexity than the best current factoring
>techniques. Similarly, no rational person should assure you that
>attacking DES or triple DES will never improve much beyond brute-force
>key-searches. However, I will humbly suggest to you we ARE a lot safer
>against those possibilities than similar risks with newer and less
>studied techniques - and that history and common sense DO give us the
>right to those basic assumptions contrary to the gloomy and highly
>unhelpful view you hold.

On the contrary: I have shown several different approaches which are
helpful for security even in an environment where we cannot assure
ourselves of the strength of any particular cipher. What is really

gloomy and unhelpful is this insistence that the only thing we can do is wait for the "experts" to certify a cipher so we can use it.

We hit on a cipher as hard as we can and then assume it to be strong and insist that we use that one cipher because it is "better tested" than anything new. The "better tested" part is probably true, but unless we know the capabilities of our Opponents, it hardly matters. We don't know how they hit, or how hard.

>A quick glance at any of the big mathematical problems in history,
>particularly the ones that are simply stated (ie the difficulty is not
>composed even partially out of obscurity - it looks more like a brick
>wall than a maze) almost always are either not solved even today, or
>were solved using techniques much more sophisticated than those
>available to those who posed the original question and first tried to
>solve it. Indeed the classical problems have typically given rise to
>entire branches of mathematics that grew out of a pursuit of that
>problem.
>
>Fermat's Theorem is the obvious example but there are others too.
>Someone more up to date with things could clarify, but I think they were
>trying to refine Andrew Wiles' proof a little to slice a couple of
>hundred pages off it ... it simply was not solved using a ruler and
>compass and the odd quadratic here and there. And yes, as I'm sure
>you're thinking, it IS possible it can be solved with a ruler and
>compass and the occasional discriminant. But most people will be happy
>to accept that that is a lot LESS likely to happen than if I just pose a
>new simply stated differential equation and state it can't be solved in
>simple terms only to have someone prove me wrong.
>
>Techniques, understanding, and formalised mathematical frameworks evolve
>- occasionally someone does throw something new and useful in and things
>accelerate for a while, but sudden breaks solving historical problems
>with simple techniques are VERY much the exception not the rule.

I doubt that the historical record applies to ciphers in the same way it does other problems. Nature is not deliberately trying to confuse and hide. Cryptography has a completely different situation.

>Let me ask the following - do you disagree with the following statement;
>"History has demonstrated time and time again, that the longer a problem
>resists the attack of academics, hobbyists, and mechanics - the
>probability the problem can be broken using simple techniques that were
>available at the time the problem was posed (or even comprehensible to
>the people of that time) decreases."

Yes, I disagree. Each cipher either can or can not be solved easily. A Boolean result is not a probability. We only get a probability when we have a wide variety of ciphers. And then of course we still do not know what that probability is.

>Occasionally someone invents a wheel, but divine beams of light are a
>lot less common than simple grunt-work and craftsmanship. This is also
>true of "our opponents" as you have a tendency to call them.
>
>> >Not at least trying cryptanalysis on a cipher is stupid which
>> >I'm sure you agree with.
>>
>> I do. But there is no one cryptanalysis. Indeed, there is no end to
>> it. But we do have to make an end before we can field anything. This
>> in itself tells us that cryptanalysis as certification is necessarily
>> incomplete.
>
>It is all probabilities and risk management. Mr Schneier will hopefully
>agree with me on that and I hope you do too (I hope anyone contributing
>to the crypto-frameworks I will have to use day-to-day agree with that
>also).

This is particularly disturbing: You do not know the probabilities, and you do not know the risk, yet you would have us manage the situation using exactly these quantities. That is mad.

I agree with a lot of handwave statements. I also take on the limits of the handwaves which are false. I am not against cryptanalysis; I think it should be used. I am against endowing it with mystical powers, and I am against the implication that this is how we know the strength of a cipher. Cryptanalysis gives us something, but not that. In particular, cryptanalysis does not really provide the confidence that others see in a "certified" result.

>Would you have us believe that all things that are not absolute are
>necessarily equal? God, this sounds like a debate on socialism all of a
>sudden - my humblest apologies [;-)

In ciphers, YES, I would have you so believe.

Ciphers are distinctly different from other areas of experience. The problem is that our Opponents operate in secrecy. That means we actually do not know when our ciphers fail. But unless we know about failure, we cannot assess risk. Yet you and most others attempt to interpret risk as we do in areas where we know the risk.

For example, we have some general feeling about the risk of driving our cars because we see failure announced on the news. Everybody knows the risk of flying because we see the disaster reported. Crypto failure is not reported, so we assume that risk is low. That is a faulty assumption. We do not know the risk. But in any security analysis we necessarily must assume the risk is real.

>> Our main problem is that cryptanalysis does NOT say that there is no
>> simpler attack. It does NOT say that a well-examined cipher is secure
>> from your kid sister. Oh, many people will offer their opinion, but
>> you won't see many such a claims in scientific papers, because there
>> we expect actual facts, as opposed to wishes, hopes, and dreams.
>
>But those claims say as much as; "we've hopefully done the best we can
>with the best techniques we have and the best people we can find, and
>this one seemed to resist our best attacks the best so we can only give
>you the best assurances we can that the best chance you have is to use
>this one".

Yes, those are the formal claims. And then we see everyone putting their eggs in the basket of a single cipher (or small fixed group of ciphers) once again. The formal claims are not really what is being transmitted: What people see is a "certified" cipher which everyone should use instead of "uncertified" ciphers. In fact it is openly argued that "uncertified" ciphers have more risk, without being able to quantify that risk. While I would hope every cipher would get as much analysis as it could get, the "certification" of one cipher does not give us what we need. All it would take is a failure of that one cipher for us to lose everything we try to protect.

>If you cannot interpret cryptanalytic conclusions in that fashion then
>you seem to miss their point.

On the contrary, if you cannot interpret the way those conclusions are mis-taken -- even in this group, even by you -- it is you who misses the point.

>I agree with Mr Schneier ... it is a race
>- our opponents (again using your phrase) get to see the best
>cryptanalytic techniques we have, and sometimes we get a peek (or a
>leak) at theirs ... we just do the best we can with what we've got - and
>history has shown that if we keep that up for a while, the chances of an
>about-turn due to some radical improvement in the theory decreases
>steadily.

I disagree with Schneier. I will agree that it is contest between cryptographer and HIDDEN cryptanalyst. But it is no race because we do not know what the hidden guys can do. This is about like calling AES a "contest," when the rules are hidden so the winner can be chosen in a smoke-filled back room. This is not to mention the fact that patented ciphers were kept out, yet another decision influenced by Schneier which just happens to benefit him. Just a coincidence.

>> Cryptanalysis does NOT give us an indication of how much effort our
>> Opponent will have to spend to break the cipher. Yet that is exactly
>> what the cryptanalytic process would like us to believe: That is why
>
>I disagree - your point of view has some merit but is no more valid than
>the polar opposite statement.

Hardly: The polar opposite does not provide a motive to alter the usual recumbent attitude and actually change the way we do business. Relying on any one cipher is a risk, and the extent of that risk is not known. Because the risk is unknown, it hardly makes sense to say that the experts have done all they can so we should trust the result.

Users should insist on having and using a wide and growing variety of ciphers. The fact is that these ciphers cannot be as well "certified" as any one cipher. But since "certification" cannot be complete, the possibility of failure even in such a cipher should not be ignored. But if one were to use that cipher in multiple ciphering along with two others (selected, say, by a random message key), we get the best of both worlds, at the cost of somewhat reduced throughput.

>If people devote their lives to keeping up
>to date with the literature and do their best to innovate and develop in
>full public-view, and their best attempts to break things fail for a
>period of time (and I'm talking about the crypto community as a whole
>here) then we CAN infer that that process represents a steadily
>increasing probability that it's not going to fall over tomorrow in some
>dramatic fashion. I do not mean that evolving cryptanalysis work
>provides increasing confidence in brand-new ciphers and what-not, rather
>that as one cipher builds up a catalogue of evolving cryptanalysis work
>against it that we DO have a decreasing probability that THAT cipher
>will fall over in show-stopper fashion.

We know no such thing. We have no idea how many attacks there may be in theory, so cannot judge how many of those we know. All we know is

that we know more than we used to, which is no probability at all.

>> we have the process of: 1) design a cipher, and 2) certify the
>> cipher by cryptanalysis. As I see it, the real opportunity for
>> cryptanalysis is as part of a dynamic and interactive cipher design
>> process, as opposed to final certification.
>
>And it currently isn't? What exactly does the open publication of
>research, countless conferences, news-groups, mail-lists, web-sites,
>open-source projects, etc amount to other than a dynamic and interactive
>process?

The usual refusal to re-analyze a corrected work.

>Also, thousands of hobbyists and professionals all doing their
>damndest to break each others ciphers gives me personally some
>confidence in the value of "standing the test of time".

There is no such standing without knowing real results. We have no
idea how many tests are made, with what background and effort, and
have no idea what the results were. This "test of time" is an
illusion.

>> Thanks. I suggest you learn it by heart if you intend to depend upon
>> cryptography.
>
>I suggest that you get a little more realistic. What do you have more
>confidence in, "NT.DLL" or an established release version of the linux
>kernel? Or IIS versus Apache? (again, speaking about versions which
>aren't acknowledged by the authors as being "beta"). And no, that
>question is not rhetorical, I'm actually interested to hear your
>response.

I have no opinion. Confidence in programs is far different from
confidence in ciphers. We can "test" programs, at least to see
whether they do what we want, whether they crash, and so on. Ciphers
are fundamentally different. We can test a cipher program to see
whether it crashes, but we cannot know if it is providing the
protection we want. We do not know if the cipher has already been
penetrated and is being read by our Opponents just as easily as by the
recipient. We do not know. And without knowing, we are unable to
assess risk, or build either confidence or trust.

>As for your continued suggestion that confidence in (relative)
>conclusions reached by noted cryptanalysts is overrated and work by
>lesser mortals unfairly disregarded. In reality I think you are wrong.
>(a) If a lesser mortal finds an improvement in cracking DES keys, they
>need only publish it to sci.crypt with the header "I think I can hack
>DES keys a bit faster ...CRYPHTML.HTM" and they will get all the attention to their
>claims they desire, and if they have the facts to back it up they
>needn't worry about anonymity.

Excuse me, but why would someone with such a breakthrough publish it
for free? Academics are paid to do their work, and paid to publish
(in fact, most professional journals expect the author's organization
to pay page fees). In a sense, academics are paid by society to give
their work away -- but exactly where is the payment for the individual
who comes up with similar advances? Why would they post it for free?

Anyone who thinks I am a greedy SOB, please feel free to look at my
pages and see the information there for free. I am not paid to do
that, nor am I compensated for "web excess bandwidth" charges for your
downloads. But, somewhere, there must be a profit to be able to
continue the work. People who do not get paid for publishing
cryptanalysis have scant motive to do it. Unfortunately, I expect
that our Opponents do indeed get paid for such work.

>(b) If someone with a track-record
>proposes a new cipher (or in my metaphor, an alteration to kernel.c in
>Linux) and someone unknown does the same, it is natural, right, and fair
>for me to regard the latter with more scepticism and the former with a
>little more of an open mind.

You are forced into a basically unscientific approach because you have
no way to measure the true strength of the designs. The very fact you
are behaving this way tells us much about whether such designs can be
trusted for what they are, or whether you would accept them being
promoted as something they really are not. You would.

>Perhaps this Darwinist philosophy is not to your liking but I'm afraid
>it fits the model. If I have a studied knowledge of shooting, am good at
>it myself, stay abreast of the most modern trends, and am widely
>respected as an expert in the field - then I am probably as good a
>person as any to suggest methods for staying out of the firing line.

But in shooting -- as in most other activities -- one knows the
result. Ciphers are fundamentally different in that one does not know
whether they are working or not.

>> This is my bit for public education.

>
>And it has been useful to provide for thoughtful debate - but I think
>you overreach to absolute conclusions to counter opposing conclusions
>that I don't think anybody is actually making.

It is obvious that people are making the conclusion that cryptanalysis is certification, for there has been no effort to construct protocols which deal with the fact that we can have no confidence in the resulting cipher.

>> I have no modern products. I do offer cryptographic consulting time,
>> and then I call it as I see it. I also own patented cryptographic
>> technology which could be useful in a wide range of ciphers.
>
>Great - perhaps if you would benefit us all (if that is your aim) by
>describing
>(a) how you made design decisions for your cryptographic technology
>(particularly with relationship to your awareness of classical and
>modern loopholes and weaknesses you were trying to avoid).

Basically I started out in stream ciphers, and read everything I could about them. As I recall, getting cryptographic information was far more difficult at the time. I followed the basic path from Vernam, and found more information about the sequence of development in the patent literature than elsewhere. We can see an ever-increasing complexity in the "running key generator" producing what I now call the "confusion sequence." As far as I could tell there had been no attempt to improve the combiner itself, and there was some feeling that a reversible nonlinear combiner was a contradiction in terms. But I did in fact find a new concept: Dynamic Substitution, which I patented and now own.

A stream cipher also needs an efficient confusion source, so I embarked on a survey of RNG technology. You can read about that in my Cryptologia article on my pages. From among the various schemes I selected the Additive RNG as being fast, and capable of expansion. For the first version, I found a primitive mod-2 polynomial of degree 11,213 and so constructed an RNG holding about 44K of state. I also innovated a new nonlinear filter to protect the RNG. The resulting CLOAK cipher used two levels of Dynamic Substitution, with 16 dynamic tables in the second level, which further protects the RNG.

With respect to my work in block ciphers, I have some descriptions of the tests I have used, and the results found, on my pages. In particular, I think the use of nonlinear complexity measurements to show the expected distribution for a larger block constructed out of smaller blocks and mixing, is fairly persuasive. Not proof, of course, but we already talked about that.

>(b) what kind of analysis has been (or could be) done on the/those
>technology(ies).

My new technologies have been ignored by academia, even when formally published in Cryptologia. Schneier has said that this is normal for patented technology. Of course, academics are compensated for the work they do; I am not.

The fact that my work is not addressed probably has negative consequences for me. But it also means that academia has no background for dealing with these structures beyond what I have personally published. That may be insufficient, but it is all there is.

>(c) how you would convince anybody that your ideas merit some degree of
>trust/faith/use/investment.

Jeez, I'm a technical guy (much to my loss I'm sure). I deliberately do not try to convince people. I do try to make information available for people to use. But as you imply most people are not able to use that information, and don't want to, but do want me to give them confidence in whatever cipher they are using. Alas, I know there is no such confidence, so I have no confidence to give them. They generally find this disturbing.

>Do you expect us to assume that even though the winning AES candidate
>will have been subjected to very deep analysis by vary many parties of
>very different angles of vested interest/disinterest, because it COULD
>be broken tomorrow it is has no more measurable "strength" than a
>boutique new idea which has not been widely distributed and tested? The
>fact two things are neither black or white does not imply they are the
>same shade of grey.

No, it implies that they have the same unknown risk: That of complete exposure. To not use one because we are afraid of that risk and then use the other which may have the same outcome is foolish.

>> I see no problem with someone promoting what they think is an advance
>> in the field, even if they will benefit. But when reasoning errors
>> are promoted which just happen to benefit one's business -- in fact, a
>> whole sub-industry -- some skepticism seems appropriate. Just once I

>> would like to see delusions promoted which produce *less* business.
>
>You call them "delusions", I call them "reasoned and qualified critiques
>open to public dissemination and review" - let's call the whole thing
>off. (as the song goes).

Which means?

Terry Ritter ritter@io.com http://www.io.com/~ritter/
Crypto Glossary http://www.io.com/~ritter/GLOSSARY.HTM

Subject: Re: Question on confidence derived from cryptanalysis.
Date: Sun, 18 Apr 1999 00:35:46 GMT
From: "Douglas A. Gwyn" <DAGwyn@null.net>
Message-ID: <371944CB.9B77BF62@null.net>
References: <3718e5e9.9093614@news.io.com>
Newsgroups: sci.crypt
Lines: 43

Terry Ritter wrote:
> It is obvious that people are making the conclusion that cryptanalysis
> is certification, for there has been no effort to construct protocols
> which deal with the fact that we can have no confidence in the
> resulting cipher.

There is a difference between "less than total confidence" and
"no confidence". In many cases, one can reliably estimate the
enemy's capabilities and be confident that he is not in as
good a position to succeed in attacks against your system as
your own tiger team is; so if your guys can't crack the system,
it is more likely than not that the enemy can't crack it either.
Depending on the potential cost of being wrong, that might be
good enough. If not, then your idea of additional protection
makes sense. However, you need to prove that your protocols
do add significant security; otherwise the new "protocolled"
system is subject to the same argument you made about the
original system, namely one doesn't know whether or not the
enemy can crack it.

> ... As far as I could tell there had been no attempt to improve
> the combiner itself, and there was some feeling that a reversible
> nonlinear combiner was a contradiction in terms.

I assume you're talking about the open literature, because it's
not the case inside the fence. That's one of the frustrating
things about this business; there is a lot of (slow) reinvention
of the wheel, due to extreme secrecy about what is known.

> The fact that my work is not addressed probably has negative
> consequences for me. But it also means that academia has no
> background for dealing with these structures beyond what I have
> personally published. That may be insufficient, but it is all
> there is.

Largely, academia studies what they already know how to study,
because the expectation of producing something "publishable"
is greater that way. This is really sad, but understandable.

Just so you know, I appreciate your work and especially your
making useful information available via the Web. Maybe self-
publication will help mankind make progress in fields that
are currently stagnating due to academic inbreeding.

Subject: Re: Question on confidence derived from cryptanalysis.
Date: Sun, 18 Apr 1999 05:20:40 GMT
From: ritter@io.com (Terry Ritter)
Message-ID: <37196b57.4012898@news.io.com>
References: <371944CB.9B77BF62@null.net>
Newsgroups: sci.crypt
Lines: 90

On Sun, 18 Apr 1999 00:35:46 GMT, in <371944CB.9B77BF62@null.net>, in
sci.crypt "Douglas A. Gwyn" <DAGwyn@null.net> wrote:

>Terry Ritter wrote:
>> It is obvious that people are making the conclusion that cryptanalysis
>> is certification, for there has been no effort to construct protocols
>> which deal with the fact that we can have no confidence in the
>> resulting cipher.
>
>There is a difference between "less than total confidence" and
>"no confidence". In many cases, one can reliably estimate the
>enemy's capabilities and be confident that he is not in as
>good a position to succeed in attacks against your system as
>your own tiger team is; so if your guys can't crack the system,
>it is more likely than not that the enemy can't crack it either.

But you know this can be skating pretty near the edge. Sometimes it's
hard to know what "we" can do (and, presumably, we *know* our guys),
let alone predict what "they" can do. But in the case of a

general-use crypto standard, this is a game we cannot play and could not afford the risk of playing anyway.

>Depending on the potential cost of being wrong, that might be
>good enough. If not, then your idea of additional protection
>makes sense. However, you need to prove that your protocols
>do add significant security; otherwise the new "protocolled"
>system is subject to the same argument you made about the
>original system, namely one doesn't know whether or not the
>enemy can crack it.

I've given the basic argument various times. The argument for multiple ciphering is well known. The argument for using many ciphers rests on partitioning the data into independently-protected channels which compartmentalizes risk by reducing single-failure loss. This also reduces the motive for Opponents who might otherwise try to attack a single standard cipher. The argument for having a continuous flow of new cipher designs rests on the cost implications to any Opponent who tries to keep up with new designs.

Of these three, the last two both depend upon having a substantial number of ciphers, and the argument is made that these cannot be as well constructed -- or as thoroughly certified -- as a single standard. My response is that we do not know, in general, that even a "certified" cipher is strong enough. If we did, we would not need all this other stuff. I think it very reasonable to leave it to each user (or security office) to select the ciphers they want to use, and modify that list based on the latest ciphers and academic results. At least that way everyone is in charge of their own fate.

>> ... As far as I could tell there had been no attempt to improve
>> the combiner itself, and there was some feeling that a reversible
>> nonlinear combiner was a contradiction in terms.
>
>I assume you're talking about the open literature, because it's
>not the case inside the fence. That's one of the frustrating
>things about this business; there is a lot of (slow) reinvention
>of the wheel, due to extreme secrecy about what is known.

Yes, of course, I know only the open literature. I have no idea what was developed otherwise. For example, after I developed Dynamic Substitution, I realized that one could use a random Latin square as a combiner. I would expect that this was long known "inside," but am unaware of any open literature about it. (Shannon of course talks about Latin squares, but does so in the context of entire cipher transformations, and not stream-cipher combiners.)

>> The fact that my work is not addressed probably has negative
>> consequences for me. But it also means that academia has no
>> background for dealing with these structures beyond what I have
>> personally published. That may be insufficient, but it is all
>> there is.
>
>Largely, academia studies what they already know how to study,
>because the expectation of producing something "publishable"
>is greater that way. This is really sad, but understandable.
>
>Just so you know, I appreciate your work and especially your
>making useful information available via the Web. Maybe self-
>publication will help mankind make progress in fields that
>are currently stagnating due to academic inbreeding.

Coming from you, that means a lot. Thanks.

Terry Ritter ritter@io.com http://www.io.com/~ritter/
Crypto Glossary http://www.io.com/~ritter/GLOSSARY.HTM

Subject: Re: Question on confidence derived from cryptanalysis.
Date: Tue, 20 Apr 1999 17:36:26 GMT
From: jsavard@tenMAPSONeerf.edmonton.ab.ca (John Savard)
Message-ID: <371cb65e.4479246@news.prosurfr.com>
References: <37196b57.4012898@news.io.com>
Newsgroups: sci.crypt
Lines: 49

ritter@io.com (Terry Ritter) wrote, in part:
>On Sun, 18 Apr 1999 00:35:46 GMT, in <371944CB.9B77BF62@null.net>, in
>sci.crypt "Douglas A. Gwyn" <DAGwyn@null.net> wrote:

>>I assume you're talking about the open literature, because it's
>>not the case inside the fence. That's one of the frustrating
>>things about this business; there is a lot of (slow) reinvention
>>of the wheel, due to extreme secrecy about what is known.

>Yes, of course, I know only the open literature. I have no idea what
>was developed otherwise. For example, after I developed Dynamic
>Substitution, I realized that one could use a random Latin square as a
>combiner. I would expect that this was long known "inside," but am
>unaware of any open literature about it. (Shannon of course talks
>about Latin squares, but does so in the context of entire cipher

>transformations, and not stream-cipher combiners.)

>>Largely, academia studies what they already know how to study,
>>because the expectation of producing something "publishable"
>>is greater that way. This is really sad, but understandable.

>>Just so you know, I appreciate your work and especially your
>>making useful information available via the Web. Maybe self-
>>publication will help mankind make progress in fields that
>>are currently stagnating due to academic inbreeding.

>Coming from you, that means a lot. Thanks.

Although lately, once again, I've made a number of posts criticizing places
where I think you've overstated your case - and I think it's very important
not to overstate one's case when one is advocating a minority position -
I will take the opportunity to acknowledge both that you have made
contributions through your own work, as well as by representing a point of
view that points in the direction of what I, also, feel is a correction
needed by the cryptographic community.

One needs the very highest credibility when one is engaged in telling
people what they do not want to hear.

As I, too, know "only what I read in the papers", I have no idea if someone
in Serbia reading my web page has forced the NSA to spend X billions of
dollars on new computers - I don't believe I've said anything in my own
designs that would not have been obvious to professionals even in countries
with far less impressive cryptographic capabilities than those of the U.S.
- but I tend to believe that that particular horse was out of the barn even
before Phil Zimmerman came along. But I could be wrong.

John Savard (teenerf<-)
http://members.xoom.com/quadibloc/index.html

Subject: Re: Question on confidence derived from cryptanalysis.
Date: Tue, 20 Apr 1999 22:03:05 GMT
From: ritter@io.com (Terry Ritter)
Message-ID: <371cf98b.7553742@news.io.com>
References: <371cb65e.4479246@news.prosurfr.com>
Newsgroups: sci.crypt
Lines: 68

On Tue, 20 Apr 1999 17:36:26 GMT, in
<371cb65e.4479246@news.prosurfr.com>, in sci.crypt
jsavard@tenMAPSONeerf.edmonton.ab.ca (John Savard) wrote:

>[...]
>Although lately, once again, I've made a number of posts criticizing places
>where I think you've overstated your case

Well, I guess in that case I need to go back and directly address
those issues. They frankly seemed less compelling than some others at
the time, and there is only so much time. In fact, I'm going to have
to finish this up soon and get back to work.

>- and I think it's very important
>_not_ to overstate one's case when one is advocating a minority position -

I think it should be very disturbing to anyone actually trying to do
Science to have to consider whether or not the conclusions they come
to are a "minority position." It really does not matter how people
vote on the facts: The facts are what they are. I do not even think
about whether my "positions" are minority or majority, and I do not
care.

I don't suppose there ever has been or ever will be anything I write
that I will not look back on and say "Gee, I could have put that
better." But I see no overstatement. If you do, you should state
clearly what you consider the limits of the correct position, and
highlight the excess which you consider beyond reality.

What I see in my comments is an attempt to correct certain irrational
conclusions about cryptanalysis and strength which may have very
significant negative consequences for society. This should be pretty
much a straight logic argument with little opinion involved. The
issue reappears periodically, but has been promoted recently in
various writings by Schneier (in particular, the article in the March
IEEE Computer column).

>I will take the opportunity to acknowledge both that you have made
>contributions through your own work, as well as by representing a point of
>view that points in the direction of what I, also, feel is a correction
>needed by the cryptographic community.

One is tempted to ask why -- if you think this correction is needed --
you are not also trying to make it happen. Will you step into the
breach as I retire? If you are not part of the solution....

>One needs the very highest credibility when one is engaged in telling
>people what they do not want to hear.

On the contrary, all one needs to do is to show the logic: It is compelling.

If people want to wait for a crypto god to find a way to gracefully change his point of view, fine, but that is rumor and superstition, not Science. To really know what is going on you have to be able to draw your own conclusions, and to believe your own results. It is not my goal to provide a different package of rumor and superstition which happens to be correct. I am no crypto god, and I don't want to be one. This is not about me.

Terry Ritter ritter@io.com http://www.io.com/~ritter/
Crypto Glossary http://www.io.com/~ritter/GLOSSARY.HTM

Subject: Re: Question on confidence derived from cryptanalysis.
Date: Sun, 18 Apr 1999 06:38:36 GMT
From: tuwatc@buwoqwbopu.jxf (ovbxotm)
Message-ID: <slrn7hiven.c9.tuwatc@tpep.nofsozwovh.yq>
References: <3718e5e9.9093614@news.io.com>
Newsgroups: sci.crypt
Lines: 5

Reading the same stuff from you Terry is getting old. Listening to you try to discuss "logic" with your tone is also getting old.

Guess its time to add an entry to my killfile.

Subject: Re: Question on confidence derived from cryptanalysis.
Date: Sun, 18 Apr 1999 18:48:37 -0400
From: Geoff Thorpe <geoff@raas.co.nz>
Message-ID: <371A6145.FAE5E8B@raas.co.nz>
References: <3718e5e9.9093614@news.io.com>
Newsgroups: sci.crypt
Lines: 570

Hello,

Terry Ritter wrote:
> >You want to sound a cautionary note that we all risk being naive and
> >over-confident in our "cryptanalytic testing" of ciphers - excellent
> >point and it is well taken.
>
> No, the point is NOT well-taken. It is ignored and brushed off as
> trivial and known. Then everyone sticks their head in the sand again
> until I bring it up again. This has happened for years.

Once again, we are in disagreement - philosophically and factually it would appear. From your postings, I can understand why you think this, but it is based on a premise I simply do not accept and will not no matter how many times you repeat it. Namely, that repeated cryptanalytic testing does not provide a measure of the tested strength of a cipher. You often repeat claims to the effect or (a caveat as I'm not using your exact words) "if you can't break it or prove it unbreakable then you know nothing of the cipher's strength".

Abstract, extremist, conspiracy-theoretic poppycock.

I don't want to devolve into word games but it seems necessary here to at least address our difference, although I doubt it will lead to a "resolution" of that difference. Namely, the meaning of "strength". Me, like I said in my previous post, I regard it as a practical measure, not a theoretical one.

An algorithm is not implicitly "strong" - it is just an algorithm. It is not strong "against attack" in any purely abstract sense (unless you can prove it so; very unlikely in most cases for now it would seem). I measure strength as a fuzzy quality with many fuzzy factors and am quite happy to do so. You seem to find this objectionable and your puristic approach, though obviously suitable for you, would be unacceptable and impractical for me. Many other people too, I dare suggest, are in the real world and have to "make calls" on such things rather than sitting around contemplating their navals and espousing puristic and entirely unhelpful messages of gloom. I prefer to be pragmatic than idealistic.

> >However, please do not go so far as to be
> >similarly naive yourself, and to play things out to an theoretical abyss
> >and expect us to follow you there.
>
> The abyss is there. By not following, you are in it.

You claim an abyss - and your justifications for it risk sending this discussion into the "There is a god - No there isn't" realm ... that is to say, our views seem axiomatically different and I don't expect one of us to substantially sway the other. If you claim there is a God, and I can't prove there isn't one, that does not imply that a God exists (for me at least). I consider triple-DES to be pretty "strong" - but you claim that we don't know how our "enemy" might be attacking it and our inability to fundamentally crack it means little - it's no more "strong"

than anything else we haven't broken yet. I don't agree but I can't PROVE you wrong. It does not mean you're right - and many people share MY point of view on this. Please do not be so arrogant to state your opinion as fact and to deride others for not agreeing.

> But the only thing being "measured" here is the open, academic
> analysis. The *real* experts do not play this way. We thus have no
> way to understand their capabilities. The strength value measured on
> academics cannot apply to the real problem.

perhaps this is because you're a little disgruntled with what you clearly see as ivory tower academics who don't measure up to the military resources or the hard-working "non-academics" of the world? Who knows. I certainly have no time for such cold-war style conspiracy theories - that somehow the knowledge present in military/intelligence agencies or foreign unstable countries (do the US still call them "commies"??) is probably so far distanced and disimilar to what is available in the open as to make work, developments, and failures (to break ciphers) in the open completely irrelevant when considering "the enemy"'s corresponding work. They do not, as far as I know, take new-borns into their underground caves and transform them into teenage mutant ninja-cryptologists. In fact if I had to make a guess, I'd probably say that "they" (to avoid the Men In Black paying me a visit) would constitute the equivalent of a strong university research department with a few hundred times the computing power (and a few thousand times the budget). They're still the same species, again - as far as I know.

I know such views are not so fashionable, but I really don't fear the NSA and co's ability to be vastly more clever in punching through theoretical walls that seem impossible to us in the public - I regard it as a much greater risk that they have hidden agendas in the work that they do, seem to have a lack of checks and balances to protect against abuses (civil liberties and other such things), and technological arsenal enabling impractical breaks (for the public) to be highly practical ones.

> On the contrary: I have shown several different approaches which are
> helpful for security even in an environment where we cannot assure
> ourselves of the strength of any particular cipher. What is really
> gloomy and unhelpful is this insistence that the only thing we can do
> is wait for the "experts" to certify a cipher so we can use it.

Exactly when did the experts certify triple-DES? If you're talking standards (or Government) committees putting a seal on it then no problem - I don't think anyone really thinks that gives the cipher an overnight guarantee of "strength". And when the AES winner is announced - I don't think anyone, including the author, will acknowledge that this means anything more than - "well, it's lasted this far so hopefully it will continue to hold out longer yet, and it seems to suit the logistical needs well - performance, ease of implementation, etc".

As for the "experts" certifying a cipher ... I've yet to see a widely referenced paper by a widely referenced author that claims a cipher to be "secure" in any sense other than, "not insecure from the point of attack of this particular piece of research". Most papers I've seen in fact continue to add in the usual syntactical caveats even when most reasonable humans would infer them automatically. In fact, lately sci.crypt has demonstrated perfectly that the only ones claiming "100% secure" are the boutique unknowns who insist on heated and irrational accusations targetted at established technologies and technologists. Generally I find that the "experts" tend to be quite cautious in their own conclusions of their own products, ideas, and research.

> We hit on a cipher as hard as we can and then assume it to be strong
> and insist that we use that one cipher because it is "better tested"
> than anything new. The "better tested" part is probably true, but
> unless we know the capabilities of our Opponents, it hardly matters.
> We don't know how they hit, or how hard.

No, but that does not mean their abilities are completely independant of ours. They probably grow up spoon-fed on the same academic literature that "we" are, continue to kept up to date with "our" developments - and may occasionally discover something before us, or even more radically - something we don't discover at all. This does not mean they exist in some parallel universe whereby our own work and conclusions can not be used to make even educated guesses as to what "they" might be able to achieve. That is simply naive and stubborn.

> I doubt that the historical record applies to ciphers in the same way
> it does other problems. Nature is not deliberately trying to confuse
> and hide. Cryptography has a completely different situation.

That is a very vague dismissal of my point - without even attempting to justify your own statement, let alone why mine might have been wrong. Mathematical problems - that is what I was referring to ... Fermat, squaring the circle, approximating Pi, you name it - the same theme arises and I think the model does say something relevant for cryptography. Namely, lots of easily stated problems have easy solutions, lots of complicated problems have easy or complicated solutions (the remainder have no solutions), and there are a few pesky problems that are easy to state but prove very difficult to break. Here I mean, that the difficulty seems to stem not from any difficulty to phrase or comprehend the question correctly, but from some intrinsic "resilience" to attack (yes, using CONVENTIONAL methods of the time).

History DOES provides an argument that the longer those pesky problems stay around, despite determined efforts to crack them - even developing entire branches of maths around them, then the probability DOES go down that someone is just going to slap their head and say "oops - damn, it was so obvious - why didn't we try that before". Sure it can happen, but like I said - we're talking probabilities and risk-management ... I don't mind if the coming years bring deep deep advances in finite algebra to the point that in 20 years, someone can break triple-DES with a polynomial-like complexity on time, key-length, and known plain-texts. But I will highly ticked off if someone discovers that despite years of cryptanalysis, it's actually easy to break it using well established techniques and we should have spotted it before (and the military already had).

> >Let me ask the following - do you disagree with the following statement;
> >"History has demonstrated time and time again, that the longer a problem
> >resists the attack of academics, hobbyists, and mechanics - the
> >probability the problem can be broken using simple techniques that were
> >available at the time the problem was posed (or even comprehensible to
> >the people of that time) decreases."
>
> Yes, I disagree. Each cipher either can or can not be solved easily.
> A Boolean result is not a probability. We only get a probability when
> we have a wide variety of ciphers. And then of course we still do not
> know what that probability is.

And here you've just completely misunderstood, I hope as an oversight and not just to be provocative. I will agree that each cipher can or can not be solved easily - depending on suitably pinned-down definitions of "solved" and "easily". And yes that represents a boolean characteristic of the algorithm (applies to implementation too). But I was talking about the probability of that characteristic being true when it has not yet been discovered and yet people have been working hard to find such an "easy" "solution". If you really don't get this, rather than you just not reading it carefully, let me wander down a STATS101 example ...

I have a coin - I can see that one side has "heads". I acknowledge that the other side could either be a "tails", or someone has slipped me a bogus coin and both sides are "heads". I will even (for the benefit of the cipher-breaking metaphor) give the coin the benefit of the doubt that most likely, the other side is a "tails". However, after flipping the coin 4 times and it landing heads each time I'm starting to get a little more confidence that someone has slipped me a bogus coin. 400 heads later I'm really beginning to feel that the coin is bogus, or I'm incredibly unlucky. However, the other side of that coin was always a head or a tail - but until we determine that the best we can get is a (maybe conditional) probability. I'm not suggesting that we now have quite the confidence in triple-DES that I would have after flipping 400 heads with my coin, but if you post a new cipher tomorrow - I WILL have the same confidence in it (or less) than if I hadn't flipped the coin yet.

> >It is all probabilities and risk management. Mr Schneier will hopefully
> >agree with me on that and I hope you do too (I hope anyone contributing
> >to the crypto-frameworks I will have to use day-to-day agree with that
> >also).
>
> This is particularly disturbing: You do not know the probabilities,
> and you do not know the risk, yet you would have us manage the
> situation using exactly these quantities. That is mad.

Do you insure your car? Do you know the probabilities or the risk? I take a look at my driving, most others' driving, the risk (=trashing the car with no insurance), the probabilities (guess work based on my driving and others' too), the cost of insurance, and make a judgement call. It's called risk management and it's also called the real world. What information do I have on a brand-new cipher? I can probably download a PDF and some source code. What information do I have on triple-DES? It's still standing. I make a judgement call - don't call me MAD for that. I think you're mad if don't see the distinction.

> I agree with a lot of handwave statements. I also take on the limits
> of the handwaves which are false. I am not against cryptanalysis; I
> think it should be used. I am against endowing it with mystical
> powers, and I am against the implication that this is how we know the
> strength of a cipher. Cryptanalysis gives us something, but not that.
> In particular, cryptanalysis does not really provide the confidence
> that others see in a "certified" result.

Mystical??? I think your sticking up your own strawmen here. Ask any implementor out there - "is IDEA breakable?" - I expect the answer in most cases to be - "dunno, it seems pretty strong". If so, they'd be using the same definition of strong I use.

> >Would you have us believe that all things that are not absolute are
> >necessarily equal? God, this sounds like a debate on socialism all of a
> >sudden - my humblest apologies [;-)
>
> In ciphers, YES, I would have you so believe.

Well I think you're wrong. But I won't have you believe anything you don't want to believe.

> Ciphers are distinctly different from other areas of experience. The
> problem is that our Opponents operate in secrecy. That means we

> actually do not know when our ciphers fail. But unless we know about
> failure, we cannot assess risk. Yet you and most others attempt to
> interpret risk as we do in areas where we know the risk.

Bull**** ... the risk is extremely well known; the risk is that someone can break our symmetric ciphers like water biscuits. We all know that risk - it's the probabilities that are open to debate. And I'm simply saying that a cipher not falling over after an extended period of review and all-out attack helps the probabilities. Wherever the probability is not 0 or 1 (or exactly 0.5) there is room for a surprise - in risk management you weigh up the probabilities with the effect of the possible outcomes, and make the best judgement call you can from that.

Me, I'm going to stick with RSA and triple-DES for a while. If you can't get a lot of worthwhile review of your technologies than that is a shame and may be doing you and your ideas a horrible disservice - but unfortunately as far as the real world is concerned, for now that DOES make your technology a bigger risk than RSA and triple-DES. Sorry but there it is.

> For example, we have some general feeling about the risk of driving
> our cars because we see failure announced on the news. Everybody

No I get a feeling of the risk because everyday I take the car out onto the road and others fail to hit me almost every time. That's how I get a general feeling for the risk.

> knows the risk of flying because we see the disaster reported. Crypto
> failure is not reported, so we assume that risk is low. That is a
> faulty assumption. We do not know the risk. But in any security
> analysis we necessarily must assume the risk is real.

Sure it's reported, as long as it is discovered by someone who reports such things. So the risk is that crypto fails, but fails in secrecy (and noone else independently reaches the same discovery and reports it). If the people who break these things without reporting it have skills completely independant of ours, or a large order of magnitude greater, then our failure to break it is independent of their failure or success. Otherwise, our failure to break it DOES decrease the chances that they have. The risk IS real, but the probability is not unrelated to our own abilities. That is just not the real world.

> Yes, those are the formal claims. And then we see everyone putting
> their eggs in the basket of a single cipher (or small fixed group of
> ciphers) once again. The formal claims are not really what is being
> transmitted: What people see is a "certified" cipher which everyone
> should use instead of "uncertified" ciphers. In fact it is openly
> argued that "uncertified" ciphers have more risk, without being able
> to quantify that risk. While I would hope every cipher would get as
> much analysis as it could get, the "certification" of one cipher does
> not give us what we need. All it would take is a failure of that one
> cipher for us to lose everything we try to protect.

Exactly why do you, or many other designers, put multiple stages in a cipher design. I'm guessing it's so that the cipher is at least as strong as the strongest element in the chain (assuming the symbolic "chain" here is serial and not parallel, otherwise someone can go around rather than through that element).

The continuum between a cipher using different cryptographic primitives, and a protocol (eg SSL) supporting multiple ciphers is purely one of packaging and patents. In fact, allowing multiple ciphers is perhaps weaker because once a cipher is broken, you need to ensure that you "switch" that cipher off ... whereas a cipher with multiple different stages means cracking one stage just weakens it a bit (and probably causes a bit of a panic to get people off that cipher before it falls down totally).

Perchance, how do you propose that extensible, scalable, and interoperable computer network systems be built around an indefinate length-list of ciphers - many having not undergone much analysis - and with all the inevitable problems of entities not agreeing on ciphers that they both have implemented. Some kind of distributed object model? But wait, you'd have to secure the underlying comms for THAT with something and that means getting people to agree once again ... Perhaps you want to bring the discussion above these petty real-world considerations?

> On the contrary, if you cannot interpret the way those conclusions are
> mis-taken -- even in this group, even by you -- it is you who misses
> the point.

Tell me where, especially if I've done it. I have higher hopes for something that has received a lot of review and is still standing than something that has not. So does nature, it's called natural selection. Pick a metaphor and run with it ... If a lion cub survives the first X months of life (low probability) then its chances of living to the age of Y improve greatly. etc etc etc.

> I disagree with Schneier. I will agree that it is contest between
> cryptographer and HIDDEN cryptanalyst. But it is no race because we
> do not know what the hidden guys can do. This is about like calling

And they are after all some alien race having developed an entire society of thought and process so vastly different to our own that our

own results (or lack of) give no indication whatsoever as to their foreign abilities?

> AES a "contest," when the rules are hidden so the winner can be chosen
> in a smoke-filled back room. This is not to mention the fact that
> patented ciphers were kept out, yet another decision influenced by
> Schneier which just happens to benefit him. Just a coincidence.

What were we discussing again? You said no matter how long a cipher stands up to public scrutiny and analysis, until it's broken or proved secure we have no more right to trust it than anything else. I disagreed. Now apparently AES is rigged??? These posts are long enough without that kind of divergence.

> >> Cryptanalysis does NOT give us an indication of how much effort our
> >> Opponent will have to spend to break the cipher. Yet that is exactly
> >> what the cryptanalytic process would like us to believe: That is why
> >
> >I disagree - your point of view has some merit but is no more valid than
> >the polar opposite statement.
>
> Hardly: The polar opposite does not provide a motive to alter the
> usual recumbent attitude and actually change the way we do business.
> Relying on any one cipher is a risk, and the extent of that risk is
> not known. Because the risk is unknown, it hardly makes sense to say
> that the experts have done all they can so we should trust the result.

So you would have us all jump from one cipher to the next, making interoperability and standardisation nigh on impossible because all out attack on a few select (and widely discussed) algorithms will tell us nothing? No thanks. This is one sure way to guarantee that "they" definately CAN break a good percentage of our traffic.

> Users should insist on having and using a wide and growing variety of
> ciphers. The fact is that these ciphers cannot be as well "certified"
> as any one cipher. But since "certification" cannot be complete, the
> possibility of failure even in such a cipher should not be ignored.

No, but sound risk management should weigh up the fact that the more homegrown, back-country, and un-analysed ciphers you employ, the more certain you can be that you're using something some of the time that can be broken without trouble. Conversely, you are right that using one simple cipher can be a risk also. However, a well designed cipher should, I hope, rely on at least a couple of stages based on some effectively independant design ideas - achieving much the same thing as stringing 2 or more independent ciphers end on end. I am not a cipher designer however so I will yield to those who are to comment further on this idea.

While we're on the subject ... it seems most crypto protocols (SSL, PKCS#7/SMIME, OpenPGP? - not sure about that one) employ a bank of ciphers. And to be honest, if say 3 ciphers get through the AES process intact and all exhibit excellent performance or implementation characteristics ... I dare say the 2 that don't "win" will still get their fair share of implementation. If this one can be optimized well for smart-cards, but that one is much better for high-throughput implementations, the industry (not Government agencies) will push its considerable weight in that direction. I just don't think anyone should use the 128-bit cipher I came up with during an episode of the X-files just because you say in theory it's as strong as triple-DES until someone breaks either one of them.

> But if one were to use that cipher in multiple ciphering along with
> two others (selected, say, by a random message key), we get the best
> of both worlds, at the cost of somewhat reduced throughput.

And this can't be achieved within ONE cipher? When you start talking multiple algorithms, you instantly start talking interoperability and standardisation headaches. You also increase the number of "pieces" in your puzzle when simplicity is far preferable. I see a security by obscurity argument in here somewhere ...

> >dramatic fashion. I do not mean that evolving cryptanalysis work
> >provides increasing confidence in brand-new ciphers and what-not, rather
> >that as one cipher builds up a catalogue of evolving cryptanalysis work
> >against it that we DO have a decreasing probability that THAT cipher
> >will fall over in show-stopper fashion.
>
> We know no such thing. We have no idea how many attacks there may be
> in theory, so cannot judge how many of those we know. All we know is
> that we know more than we used to, which is no probability at all.

wrong. We know that existing attacks have failed to bust that cipher so far, and we know how much time/effort it stood up to. Let's assume (reasonably) that "the enemy" is privvy to all our documented techniques - then, what we know forms part of their arsenal. Then, we know that THAT proportion of their arsenal has been failing for some time to break the cipher too. This gives us better chances than if we're not even being sure if our own could break it with a little time and effort.

> >And it currently isn't? What exactly does the open publication of
> >research, countless conferences, news-groups, mail-lists, web-sites,
> >open-source projects, etc amount to other than a dynamic and interactive
> >process?
>

> The usual refusal to re-analyze a corrected work.

You sound bitter. Please answer the question with some explanation, justification, or even a reference - or move on ... perhaps you think that because the (US?) Government runs everything that political rather than industrial considerations pave the way? Well, I have a lot more faith in the industry and innovative people than you do if that's the case.

> >Also, thousands of hobbyists and professionals all doing their
> >damndest to break each others ciphers gives me personally some
> >confidence in the value of "standing the test of time".
>
> There is no such standing without knowing real results. We have no
> idea how many tests are made, with what background and effort, and
> have no idea what the results were. This "test of time" is an
> illusion.

I see ... so the mystical men in black theory, put forward without evidence, should be allowed to dictate our thinking? All things, including ciphers, are relative. For the purposes of this post (now quite a huge one) I really don't care any more if "they" (the spooky people) have broken anything or not ... the fact is that on our side of the fence we've got reason to rate certain ciphers as having been tested more rigorously than others, and that (in lieu of ANY useful information about the spooky people) is what I intend to use in my decision making.

> protection we want. We do not know if the cipher has already been
> penetrated and is being read by our Opponents just as easily as by the
> recipient. We do not know. And without knowing, we are unable to
> assess risk, or build either confidence or trust.

translation: "Without knowing if it is true or false, we cannot assess the probability as to whether it is true or false".

> You are forced into a basically unscientific approach because you have
> no way to measure the true strength of the designs. The very fact you
> are behaving this way tells us much about whether such designs can be
> trusted for what they are, or whether you would accept them being
> promoted as something they really are not. You would.

Who's talking about "TRUE STRENGTH" ... we already agree that until it's proved secure or broken that we can't measure THAT, if in fact THAT exists at all. If I'm being forced into a basically unscientific approach - fine, I'm going for a pragmatic one instead - I'm talking about "tested strength". You on the other hand would prefer to run away from the issue and give no value to vast existing cryptanalytic work on widely distributed ciphers because "the enemy might have already have broken them". I simply do not think that's rational.

> >Perhaps this Darwinist philosophy is not to your liking but I'm afraid
> >it fits the model. If I have a studied knowledge of shooting, am good at
> >it myself, stay abreast of the most modern trends, and am widely
> >respected as an expert in the field - then I am probably as good a
> >person as any to suggest methods for staying out of the firing line.
>
> But in shooting -- as in most other activities -- one knows the
> result. Ciphers are fundamentally different in that one does not know
> whether they are working or not.

alteration: If for any key, it encrypts and can successfully decrypt, then it is working. What we don't know is if someone "else" has broken it, but "we" haven't yet. Unless you are overly paranoid, it is not unreasonable to draw "probabilistic" conclusions relating the "their" abilities and "ours". My point is even more straightforward than that - if our people can break it, then of course they can too; if ours try but can't, that improves our chances a little that "they" haven't. Not even knowing whether "we" can crack it after a period of time is just opening the probabilistic window wider than we should for anything we plan to use.

> It is obvious that people are making the conclusion that cryptanalysis
> is certification, for there has been no effort to construct protocols
> which deal with the fact that we can have no confidence in the
> resulting cipher.

Well if I had NO confidence in the cipher, why would I be using it? I've got loads of compressors and encoders I can call upon, why would I use a cipher if I have no confidence in it doing it's job? Presumably, any such constructed protocol would provide a safeguard against a cipher not doing its job, that is encrypting with some degree of confidence. In other words, your protocol would be a security protocol. Do you see anything at all recursive here or is it just me?

[snipped lots of good stuff about your technologies, which I liked and do not have any beef with at all]

> >(b) what kind of analysis has been (or could be) done on the/those
> >technology(ies).
>
> My new technologies have been ignored by academia, even when formally
> published in Cryptologia. Schneier has said that this is normal for
> patented technology. Of course, academics are compensated for the
> work they do; I am not.

Well, you'll have to settle that with him and the others if you can. Like I said earlier, this may be doing you and your ideas a great disservice, but as long as it stands that way - people DO have the right to regard your ideas as "riskier" than the "rusted but not busted" ones.

> The fact that my work is not addressed probably has negative
> consequences for me. But it also means that academia has no
> background for dealing with these structures beyond what I have
> personally published. That may be insufficient, but it is all there
> is.

So all ciphers are innocent until proven guilty? Unfortunately, when people's privacy and identity are at stake, ciphers (and other cryptographic primitives) are guilty until the prosecution have failed time and time again to get a conviction. It gets even worse, the cipher is never truly innocent, it just has a slowly decreasing degree of suspicion surrounding it.

> No, it implies that they have the same unknown risk: That of complete
> exposure. To not use one because we are afraid of that risk and then
> use the other which may have the same outcome is foolish.

So what are we to do? Anyway, are we talking here about the chances of a cipher getting busted (ie the whole "strength" issue), or about the effect it would have if it DOES get busted. Whatever you use (be it 3 "ciphers" strung in a line), call it a cipher and go back to square one of the problem. If you keep changing ciphers, then you and I (and you and everybody else) will not have interoperating systems.

> >You call them "delusions", I call them "reasoned and qualified critiques
> >open to public dissemination and review" - let's call the whole thing
> >off. (as the song goes).
>
> Which means?

One man's trash is another man's treasure ... insert any vaguely similar cliche for the same effect. I think repeated attempts by many people to break something and failing represents "tested strength". You think it represents "delusions". What are we to do?

Cheers,
Geoff

Subject: Re: Question on confidence derived from cryptanalysis.
Date: Mon, 19 Apr 1999 05:16:48 GMT
From: "Douglas A. Gwyn" <DAGwyn@null.net>
Message-ID: <371AD828.30BA797A@null.net>
References: <371A6145.FAE5E8B@raas.co.nz>
Newsgroups: sci.crypt
Lines: 99

Geoff Thorpe wrote:
> perhaps this is because you're a little disgruntled with what you
> clearly see as ivory tower academics who don't measure up to the
> military resources or the hard-working "non-academics" of the world?
> Who knows. I certainly have no time for such cold-war style
> conspiracy theories - that somehow the knowledge present in
> military/intelligence agencies or foreign unstable countries (do the
> US still call them "commies"??) is probably so far distanced and
> disimilar to what is available in the open as to make work,
> developments, and failures (to break ciphers) in the open completely
> irrelevant when considering "the enemy"'s corresponding work.

It depends on who you conceive the potential "enemy" to be. For many perfectly decent people, their own governments can become their enemies; history gives us many instances of this.

While the cryptanalytic bureaus of most third-world countries might not be very advanced, the one in the US certainly is. Therefore, it is quite appropriate to be concerned with *provable* security instead of "nobody in academia has a clue" security. The former should, if properly applied, stand up against *any* enemy, while the latter stands up only against, shall we say, amateurs.

(I'm not suggesting that academics haven't made useful contributions to the state of the art, just that their work does not define the total state of the art.)

> Mathematical problems - that is what I was referring to ...

Yes, cryptology is largely applied mathematics, but practical cryptanalysis has evolved in the context of operational experience that is largely unavailable to outsiders, and that has caused a substantial difference between insiders and outsiders in their aims and methods.

Some problems, like efficient factoring, are obviously relevant, and unlikely to be achieved in secret without happening in the outside around the same time. Other breakthroughs have been kept secret for decades, in some cases. So there really is reason to fear that the most advanced "enemies" might know how to easily crack some

system you use that appears uncrackable to all outsiders.

> But I will highly ticked off if someone discovers that despite
> years of cryptanalysis, it's actually easy to break [3DES] using
> well established techniques and we should have spotted it before
> (and the military already had).

There is a significant difference between what is "well established" in long-existing, well-funded cryptologic dispersed, high-turnover-rate academic community.

There is a big problem in working in *applied* fields academically, since it is harder to get academic respect from publication of application or tutorial papers instead of research papers. There are many technologies that are well-known *in general* in the research community, but their specific application to cryptology is *not* well known.

> We all know that risk - it's the probabilities that are open to
> debate. ...

More precisely, the likelihoods.
The nice thing is that *relative* likelihoods can be estimated and used to make decisions; e.g. "I need a cipher that <meets certain requirements> -- pick one."
If the consequences of not making a decision are sufficiently severe, then even an uncertain decision can be better than letting the uncertainty stop you from making a decision.

> Me, I'm going to stick with RSA and triple-DES for a while.

In a well-designed cryptosystem, these do seem sufficiently secure against realistic threats for the near future. Any vulnerabilities would most likely occur elsewhere in the system/protocols, not in these encryption algorithms as such (assuming of course a long RSA key, and 168-bit 3DES key).

> Exactly why do you, or many other designers, put multiple
> stages in a cipher design. I'm guessing it's so that the
> cipher is at least as strong as the strongest element in
> the chain ...

That seems to be part of Ritter's aim, but others seem to think that during cryptanalysis the stages have to be peeled like an onion, and they assume that there is not enough pattern available at the next-to-outermost layer for there to be any chance of peeling the outer layer off.

> And this can't be achieved within ONE cipher? When you start talking
> multiple algorithms, you instantly start talking interoperability
> and standardisation headaches.

That's a significant concern, because breakdowns in operational procedure often provide the enemy analyst the entering wedge he needs to crack a system.

Subject: Re: Question on confidence derived from cryptanalysis.
Date: Mon, 19 Apr 1999 13:37:32 -0400
From: Geoff Thorpe <geoff@raas.co.nz>
Message-ID: <371B69DC.17EE09E1@raas.co.nz>
References: <371AD828.30BA797A@null.net>
Newsgroups: sci.crypt
Lines: 195

Hello,

"Douglas A. Gwyn" wrote:
> It depends on who you conceive the potential "enemy" to be.
> For many perfectly decent people, their own governments can
> become their enemies; history gives us many instances of this.

Of course, and for most people following the current crypto issues even passively, I think they regard the regulatory and military arms of government to be the biggest problem.

> While the cryptanalytic bureaus of most third-world countries
> might not be very advanced, the one in the US certainly is.

Agreed - it would extraordinarily naive to dispute that fact. The point I was trying to make was that the collective academic grunt (and other "in the open" contributors) we have in cryptography and cryptology does not (or rather, can not) pale so completely by comparison to "the enemy" that our research and results give no indication to a cipher's susceptibility to "theirs". Mr Ritter seemed to have very different and quite extreme view on this point. However, I get the impression you tend to agree - if we can't punch a hole in it, that lowers the odds that they can (as compared to not having really seen if WE can yet).

> Therefore, it is quite appropriate to be concerned with
> *provable* security instead of "nobody in academia has a
> clue" security. The former should, if properly applied,
> stand up against *any* enemy, while the latter stands up

> only against, shall we say, amateurs.

Provable security is a very hairy branch of science - unless you pin yourself to some pretty broad axiomatic presumptions (which themselves then become the target of much scepticism and debate) proving security becomes highly awkward. I guess this is necessary because it is nearly impossible to categorise the class of "attacks" in any meaningful way (except perhaps invoking Turing machines?! [;-) If one could show that a given cipher (key) can or can not be broken at a expected running time better than 50% of a key-space search by an appropriate Turing machine it would be quite a piece of work.

> (I'm not suggesting that academics haven't made useful
> contributions to the state of the art, just that their
> work does not define the total state of the art.)

Oh I agree completely - I was just taking issue with what I perceived to be the following idea: Until we actually break it, or *prove* it secure, we have no more measure of strength for it than for another (less "investigated") one. I feel that quite the opposite is true - it IS a very appropriate statistical measure of strength, and moreover the only realistic one we have to work with. If the total state of the art stays roughly in sync with the academics, albeit "they" may have a couple of things up their sleeves and they may often get the jump on us by a few months/years with various developments, then we can make reasoned guestimations on the strength of a cipher against them based on the strength of a cipher against us.

> > Mathematical problems - that is what I was referring to ...
>
> Yes, cryptology is largely applied mathematics, but
> practical cryptanalysis has evolved in the context of
> operational experience that is largely unavailable to
> outsiders, and that has caused a substantial difference
> between insiders and outsiders in their aims and methods.

Well yes and no ... applied mathematics has a handy of way of pushing things along nicely particularly in the area of computation (complexity) problems. "Their" ideals may be different to ours but I doubt their aims or methods are ... everybody would love to break an established cipher (with the possible exception of the patent holder), everybody would love to *prove* a cipher secure (with the possible exception of the patent holder's competition). I dare say the NSA et al have less motivation to chase down some of the more daunting theoretical possibilities for weaknesses in algorithms, especially when in reality, so many of them lead nowhere or to advances that are at best - theoretical.

"They" have budgets (albeit big ones) and they probably have things they'd rather spend it on (satelites, lobbying, hardware, breaking implementations, breaking installations, etc). OTOH, having been post-grad in a mathematics department before I know very well that this obsession for looking in every nook and cranny of all things theoretical is exactly the sort of thing academics get off on. Cracking ciphers (ie. the actual algorithm itself, not the practical implementation details that may be vulnerable) is much more the meat and veg of academics who like to play and write papers. "They" just want information, and I'm guessing just do whatever they got to do to get it - and searching endlessly for little theoretical weaknesses is probably not their top priority. That's not to say they don't do it and do it very well, but I doubt their considerable advantages in resources are put so much to this task as to make our abilities so incomparable or unrelated as some might believe.

> Some problems, like efficient factoring, are obviously
> relevant, and unlikely to be achieved in secret without
> happening in the outside around the same time. Other

I agree but I doubt very much Mr Ritter does.

> breakthroughs have been kept secret for decades, in some
> cases. So there really is reason to fear that the most
> advanced "enemies" might know how to easily crack some
> system you use that appears uncrackable to all outsiders.

I know - and there's a lot of targets out there so the odds are on that at least one of them has fallen completely to an "unpublished" source without our knowing it. However, I just think it's more likely to be something less well analysed in the "open" than something well analysed in the "open" for the reasons I've mentioned, and that Mr Ritter doesn't agree with.

On a related note (and all IMHO), bit-twiddling little ciphers are no less "mathematical" than effecient factoring. Discrete maths actually finds that cute little "permutation stuff" quite fashionable from my limited contact with it (and them). Factoring tends to interest (again from my limited contact) more the applied heads - meaning I'd give better odds to developments in faster optimizations on 64-bit platforms with super-dooper cache, than to fundamental breaks in the factoring algorithms [;-)

> There is a significant difference between what is "well
> established" in long-existing, well-funded cryptologic
> organizations and what is "well established" in the
> dispersed, high-turnover-rate academic community.

True - and to agree with Mr Ritter for a moment, I think perhaps another risk is that the academics tend to show interest in things that interest them - where as the well-funded organisations you speak of more likely show interest in things that give them the best chance of accomplishing some objective. However, this pragmatic mind-set, albeit fearsome in many ways, might give us some hope that in the ethereal hights of trying (and hoping) to break an already well-studied algorithm, they probably are less hopeful, less obsessed, and more practical and realistic. After all, RSA, IDEA may be perfect but if Win95's TCP allows a password-sniffer to leak into your PC "they" have accomplished their objective and "broken" PGP as far as "they" are concerned.

> There is a big problem in working in *applied* fields
> academically, since it is harder to get academic
> respect from publication of application or tutorial
> papers instead of research papers. There are many
> technologies that are well-known *in general* in the
> research community, but their specific application to
> cryptology is *not* well known.

Probably quite true.

> > We all know that risk - it's the probabilities that are open to
> > debate. ...
>
> More precisely, the likelihoods.
> The nice thing is that *relative* likelihoods can be estimated
> and used to make decisions; e.g. "I need a cipher that <meets
> certain requirements> -- pick one."
> If the consequences of not making a decision are sufficiently
> severe, then even an uncertain decision can be better than
> letting the uncertainty stop you from making a decision.

Exactly, and well said.

> > Me, I'm going to stick with RSA and triple-DES for a while.
>
> In a well-designed cryptosystem, these do seem sufficiently
> secure against realistic threats for the near future. Any
> vulnerabilities would most likely occur elsewhere in the
> system/protocols, not in these encryption algorithms as such
> (assuming of course a long RSA key, and 168-bit 3DES key).

I think that too, but as Mr Ritter might say - you are already in the abyss and are naive if you think that. If that is so, I am comfortable in my naivety.

> That seems to be part of Ritter's aim, but others seem to
> think that during cryptanalysis the stages have to be peeled
> like an onion, and they assume that there is not enough
> pattern available at the next-to-outermost layer for there
> to be any chance of peeling the outer layer off.

Well hopefully someone will look at this, and demonstrate some success from it. Results speak for themselves, even to ivory tower academics [;-)

> > And this can't be achieved within ONE cipher? When you start talking
> > multiple algorithms, you instantly start talking interoperability
> > and standardisation headaches.
>
> That's a significant concern, because breakdowns in operational
> procedure often provide the enemy analyst the entering wedge he
> needs to crack a system.

Exactly, and if I resort to using a different cipher every week ... the cryptanalysts will not keep up with them satisfactorily and I have a lot more confidence that "they" WILL be breaking my traffic on a semi-regular basis.

Cheers,
Geoff

Subject: Re: Question on confidence derived from cryptanalysis.
Date: Mon, 19 Apr 1999 19:05:13 GMT
From: ritter@io.com (Terry Ritter)
Message-ID: <371b7e4f.6941446@news.io.com>
References: <371B69DC.17EE09E1@raas.co.nz>
Newsgroups: sci.crypt
Lines: 169

On Mon, 19 Apr 1999 13:37:32 -0400, in <371B69DC.17EE09E1@raas.co.nz>, in sci.crypt Geoff Thorpe <geoff@raas.co.nz> wrote:

>[...]
>Agreed - it would extraordinarily naive to dispute that fact. The point
>I was trying to make was that the collective academic grunt (and other
>"in the open" contributors) we have in cryptography and cryptology does
>not (or rather, can not) pale so completely by comparison to "the enemy"
>that our research and results give no indication to a cipher's
>susceptibility to "theirs". Mr Ritter seemed to have very different and
>quite extreme view on this point. However, I get the impression you tend

>to agree - if we can't punch a hole in it, that lowers the odds that
>they can (as compared to not having really seen if WE can yet).

If you really want to bait Mr. Ritter, I'll go one round with you:

I have just previously covered my main argument, which is basically that IN MY OPINION, with a single standard cipher, there will be far too much value at risk to endure even a small possibility of single-cipher failure. I note that the POSSIBILITY of such failure is fact, not opinion. The opinion part of this is my judgment of the costs and consequences of failure, versus the additional cost of protecting against such failure.

My position is that the consequences of failure of a universal single-cipher system would be catastrophic, and that even a small probability of such failure is unacceptable. This means we cannot depend on any single cipher, no matter how well reviewed. We can reduce the probability of single-cipher failure, and reduce also the value of information at risk from any failure, by changing what we consider a cipher system to be.

What I call the cul-de-sac extension of this argument is the question of just *how* small the probability of failure is.

1) I dispute the idea that by applying various attacks to a cipher we somehow can predict how it will perform on future unknown and potentially unrelated attacks. (And if this were true, we should be able to see the effect with respect to past ciphers. This should be measurable and quantifiable in a scientific sense. But we have no such reports.)

2) I dispute the idea that by looking at the attacks we have we can somehow estimate the probability that unknown attacks exist. (Again, were this true, we should have scientific evidence to support it. But we do not.)

3) I dispute that we can estimate the capabilities of our Opponents from the capabilities we see in academics or that we can extrapolate from our open experience to predict the capabilities of our Opponents.

(Alas, there is no evidence to be had here.)

In summary: 1) We cannot estimate the probability that an effective attack exists which we did not find; and 2) We cannot estimate the probability that even if such an attack does exist, our Opponents can find it and use it. I thus claim that we CAN know nothing of the probability of future cipher failure, and cannot even reason that this probability is "small." The practical consequence of this is that we cannot trust any cipher.

IF we were willing to assume that our Opponents would use only the attacks we know and have tried, presumably we *could* have insight into the amount of effort needed to break a cipher (although we might have screwed up in testing). But I am of the opinion that we cannot assume that our Opponents have our limitations. Indeed, I think this is very basic cryptography.

>[...] I was just taking issue with what I perceived to
>be the following idea: Until we actually break it, or *prove* it secure,
>we have no more measure of strength for it than for another (less
>"investigated") one. I feel that quite the opposite is true - it IS a
>very appropriate statistical measure of strength, and moreover the only
>realistic one we have to work with. If the total state of the art stays
>roughly in sync with the academics, albeit "they" may have a couple of
>things up their sleeves and they may often get the jump on us by a few
>months/years with various developments, then we can make reasoned
>guestimations on the strength of a cipher against them based on the
>strength of a cipher against us.

And upon what evidence do you base you opinion that we *can* predict what our Opponents can do?

Do you even have evidence that we can predict what *our* guys can do?

>[...]
>> Some problems, like efficient factoring, are obviously
>> relevant, and unlikely to be achieved in secret without
>> happening in the outside around the same time. Other
>
>I agree but I doubt very much Mr Ritter does.

The idea that *any* cipher *may* have an effective attack is fact, not opinion. The only opinion here is whether the issue is worth addressing.

Presumably, you would handwave about what our Opponents can do both now and in the future and say that caution is silly. But that conclusion is based on your opinion that we can predict what others may do in the future, which I find very strange. If that were true in general, we could put criminals in jail before they did anything.

>> breakthroughs have been kept secret for decades, in some

>> cases. So there really is reason to fear that the most
>> advanced "enemies" might know how to easily crack some
>> system you use that appears uncrackable to all outsiders.
>
>I know - and there's a lot of targets out there so the odds are on that
>at least one of them has fallen completely to an "unpublished" source
>without our knowing it. However, I just think it's more likely to be
>something less well analysed in the "open" than something well analysed
>in the "open" for the reasons I've mentioned, and that Mr Ritter doesn't
>agree with.

Mr. Ritter has always recommended that we get as much cryptanalysis as
we can. But he also points out that this is an open-ended process
which in any case must be terminated to have a product. So our
cryptanalysis can never be complete.

With respect to the problem of potential catastrophic failure from a
single-cipher system, no amount of cryptanalysis can prevent such
failure. Both untested ciphers and massively-tested ciphers are the
same in the sense that neither can be trusted.

>[...]
>> > Me, I'm going to stick with RSA and triple-DES for a while.
>>
>> In a well-designed cryptosystem, these do seem sufficiently
>> secure against realistic threats for the near future. Any
>> vulnerabilities would most likely occur elsewhere in the
>> system/protocols, not in these encryption algorithms as such
>> (assuming of course a long RSA key, and 168-bit 3DES key).
>
>I think that too, but as Mr Ritter might say - you are already in the
>abyss and are naive if you think that. If that is so, I am comfortable
>in my naivety.

Mr. Ritter would say that you are vulnerable to a single-cipher
failure. And as long as the problem is just you, we really don't
care. But if the problem eventually becomes the whole society pretty
much using the same cipher, we may care, yet be well past the time to
do much about it.

>[...]
>Exactly, and if I resort to using a different cipher every week ... the
>cryptanalysts will not keep up with them satisfactorily and I have a lot
>more confidence that "they" WILL be breaking my traffic on a
>semi-regular basis.

The whole point of that particular approach is that cryptanalysts will
not keep up. In particular, the other side will not keep up, and
those are the guys we have to worry about.

It should be possible for a true cipher designer to use various
alternatives to achieve a similar result, thus mixing and matching and
producing various different ciphers with similar supposed strength,
whatever that may be. We cannot hope to know that strength by
cryptanalysis, of course.

Terry Ritter ritter@io.com http://www.io.com/~ritter/
Crypto Glossary http://www.io.com/~ritter/GLOSSARY.HTM

Subject: Re: Question on confidence derived from cryptanalysis.
Date: Wed, 21 Apr 1999 19:07:13 GMT
From: aardwolf@telusplanet.net (Earth Wolf)
Message-ID: <371b711c.6682075@news.calgary.telusplanet.net>
References: <371b7e4f.6941446@news.io.com>
Newsgroups: sci.crypt
Lines: 110

On Mon, 19 Apr 1999 19:05:13 GMT, ritter@io.com (Terry Ritter) wrote:

>I have just previously covered my main argument, which is basically
>that IN MY OPINION, with a single standard cipher, there will be far
>too much value at risk to endure even a small possibility of
>single-cipher failure.

Depends on what you're trying to protect. Is it tactical or strategic
information? i.e. does it have to be kept secret for the next
millennium, or will it be public knowledge at 9:00 a.m. next Tuesday?
If this secret is revealed, will I lose a hundred dollars on the stock
market, or will the world be sucked into a black hole? Is my quest for
the "ultimate" cryptosystem going to be so horrendously cumbersome
that the users will refuse to use it? Is the decrease in bandwidth
going to prevent vital information from being disseminated in a timely
fashion?

Remember Pearl Harbour, where warning of the impending attack arrived
by messenger amid the smoking aftermath? A clear case of how too much
security can be almost as bad as none at all. Real world security
deals with these kinds of trade-offs in a way that your ivory tower
thinking can never comprehend, I'm afraid.

>1) I dispute the idea that by applying various attacks to a cipher we
>somehow can predict how it will perform on future unknown and
>potentially unrelated attacks. (And if this were true, we should be
>able to see the effect with respect to past ciphers. This should be
>measurable and quantifiable in a scientific sense. But we have no
>such reports.)

What kinds of reports are you looking for? There are lots of archaic ciphers which were considered unbreakable in their day which are child's play to solve with modern technology. The Jefferson wheel, for example. What more were you looking for?

>In summary: 1) We cannot estimate the probability that an effective
>attack exists which we did not find;

Of course we can. I estimate it to be 17.375%. It may not be the most reliable estimate, of course :-)

>I thus claim that we CAN know nothing of the
>probability of future cipher failure, and cannot even reason that this
>probability is "small." The practical consequence of this is that we
>cannot trust any cipher.

I'll trust DES a heck of a lot more than I trust ROT-13. And I'll trust 3DES a heck of a lot more than I trust DES.

>
>IF we were willing to assume that our Opponents would use only the
>attacks we know and have tried, presumably we *could* have insight
>into the amount of effort needed to break a cipher (although we might
>have screwed up in testing). But I am of the opinion that we cannot
>assume that our Opponents have our limitations. Indeed, I think this
>is very basic cryptography.

No, basic cryptography involves making your best estimate of your opponents' capabilities and desiging a cipher which, to the best of your knowledge, will be impervious to those capabilities for as long as it needs to be.

>And upon what evidence do you base you opinion that we *can* predict
>what our Opponents can do?

Basically, the same way we can predict what the surface temperature is on Mercury, or anything else that cannot be measured directly. We take what observations we can and attempt to extrapolate what we *don't* know.from what we *do* know.

That's basic physics, btw. :-)

>Presumably, you would handwave about what our Opponents can do both
>now and in the future and say that caution is silly. But that
>conclusion is based on your opinion that we can predict what others
>may do in the future, which I find very strange. If that were true in
>general, we could put criminals in jail before they did anything.

This author makes no distinction between being able to predict something with 100% accuracy and being able to predict something with lesser accuracy. For example, magician and card-sharp John Scarne once described playing gin rummy (for money) with a player who, after shuffling the cards, would square up the deck with the bottom facing towards him. An innocent-seeming idiosyncracy, except that he now knew the bottom card in the deck (which in gin rummy never comes into play). Suppose this card were the 8 of hearts; the player cannot predict, with 100% accuracy, what the next card in the deck will be, but he knows it will not be the 8 of hearts. This seemingly insignificant piece of information gives him a huge advantage; he knows that there is little percentage in trying to fill a meld of 8's or a run of 6-7-8 or 8-9-T of hearts, and none whatsoever in trying to fill an inside run of 7-8-9.

In studying PRBGs, ability to predict the next bit with a probability of 0.5 + epsilon, where epsilon is a small number (usually on the order of 1/polynomial(log n)) can be a huge advantage.

>With respect to the problem of potential catastrophic failure from a
>single-cipher system, no amount of cryptanalysis can prevent such
>failure. Both untested ciphers and massively-tested ciphers are the
>same in the sense that neither can be trusted.

Rubbish. I don't trust Charlie the counterfeiter, Ernie the embezzler, Rocco the rapist, or Sammy the serial killer. But I can give you a rough estimate of who I'd *least* like to crash my sister's slumber party.

Earth Wolf

Subject: Re: Question on confidence derived from cryptanalysis.
Date: Wed, 21 Apr 1999 21:33:47 GMT
From: ritter@io.com (Terry Ritter)
Message-ID: <371e4437.15431777@news.io.com>
References: <371b711c.6682075@news.calgary.telusplanet.net>

Newsgroups: sci.crypt
Lines: 210

On Wed, 21 Apr 1999 19:07:13 GMT, in
<371b711c.6682075@news.calgary.telusplanet.net>, in sci.crypt
aardwolf@telusplanet.net (Earth Wolf) wrote:

>On Mon, 19 Apr 1999 19:05:13 GMT, ritter@io.com (Terry Ritter) wrote:
>
>>I have just previously covered my main argument, which is basically
>>that IN MY OPINION, with a single standard cipher, there will be far
>>too much value at risk to endure even a small possibility of
>>single-cipher failure.
>
>Depends on what you're trying to protect. Is it tactical or strategic
>information? i.e. does it have to be kept secret for the next
>millennium, or will it be public knowledge at 9:00 a.m. next Tuesday?
>If this secret is revealed, will I lose a hundred dollars on the stock
>market, or will the world be sucked into a black hole?

I guess "yes." If there is just one standard cipher, the issue is not
so much what any one of us has to lose as it is what society as a
whole has to lose.

>Is my quest for
>the "ultimate" cryptosystem going to be so horrendously cumbersome
>that the users will refuse to use it? Is the decrease in bandwidth
>going to prevent vital information from being disseminated in a timely
>fashion?

I see no particular reason why good cryptography cannot be relatively
efficient. There might be some control overhead which might average 5
or 10 percent. Some particular ciphers might well elect to insert
"null's" to a greater extent that we have seen. But if users can
select ciphers, they can choose to not select those which have the
problems that matter to them.

>Remember Pearl Harbour, where warning of the impending attack arrived
>by messenger amid the smoking aftermath? A clear case of how too much
>security can be almost as bad as none at all. Real world security
>deals with these kinds of trade-offs in a way that your ivory tower
>thinking can never comprehend, I'm afraid.

I see a future in which most cryptography is mostly hidden, and is a
minor overhead to communications. I see homes and businesses in which
every wall switch and every lamp control is a networked device. And
if we want to control those from the Internet, all of that stuff will
need good crypto. Every light switch.

>>1) I dispute the idea that by applying various attacks to a cipher we
>>somehow can predict how it will perform on future unknown and
>>potentially unrelated attacks. (And if this were true, we should be
>>able to see the effect with respect to past ciphers. This should be
>>measurable and quantifiable in a scientific sense. But we have no
>>such reports.)
>
>What kinds of reports are you looking for? There are lots of archaic
>ciphers which were considered unbreakable in their day which are
>child's play to solve with modern technology. The Jefferson wheel, for
>example. What more were you looking for?

The question is not what *I* am looking for. My position is that no
rational extrapolation of past tests to future strength is possible.
The lack of literature containing such a thesis is consistant with my
position, and inconsistant with the alternative.

>>In summary: 1) We cannot estimate the probability that an effective
>>attack exists which we did not find;
>
>Of course we can. I estimate it to be 17.375%. It may not be the most
>reliable estimate, of course :-)

Yes. Quite amusing.

>>I thus claim that we CAN know nothing of the
>>probability of future cipher failure, and cannot even reason that this
>>probability is "small." The practical consequence of this is that we
>>cannot trust any cipher.
>
>I'll trust DES a heck of a lot more than I trust ROT-13. And I'll
>trust 3DES a heck of a lot more than I trust DES.

You are free to do as you will, including your own interpretation of
trust. However, I suspect that your meaning of "trust" for
cryptography will differ than the "trust" of other things.

My guess would be that you "trust" DES because nobody has openly
demonstrated that they can break it. So if you worry that your
information will be stolen by academics, you *can* have some
reasonable degree of trust in DES.

But if you use cryptography to protect your information from those who operate in secret and hide their successes, you have no data upon which to base trust. As Savard has pointed out, these people cannot be *less* capable than academics (unless they cannot read); that means it is *quite* likely that they are indeed *more* capable. Since you can have no published experience to guide you on the risk of using DES in such an environment, how will you gain any "trust" in it at all?

>>IF we were willing to assume that our Opponents would use only the
>>attacks we know and have tried, presumably we *could* have insight
>>into the amount of effort needed to break a cipher (although we might
>>have screwed up in testing). But I am of the opinion that we cannot
>>assume that our Opponents have our limitations. Indeed, I think this
>>is very basic cryptography.
>
>No, basic cryptography involves making your best estimate of your
>opponents' capabilities and desiging a cipher which, to the best of
>your knowledge, will be impervious to those capabilities for as long
>as it needs to be.

No, that is basic *military* cryptography, where we have known opponents and can better estimate both the probability and consequences of cipher failure.

Basic *social* cryptography (for lack of a better term) must concern itself with every non-military use for hiding data. Much of this will be financial and industrial data which is as much or more of a part of the strength of society than pure military power. Those who might attack such data are quite diverse, each with their own motives. And the consequences of a successful attack could be almost universal.

From this I conclude that the use of a single standard cipher throughout society would be an unthinkable risk.

>>And upon what evidence do you base you opinion that we *can* predict
>>what our Opponents can do?
>
>Basically, the same way we can predict what the surface temperature is
>on Mercury, or anything else that cannot be measured directly. We take
>what observations we can and attempt to extrapolate what we *don't*
>know.from what we *do* know.

And this is the same sort of answer we have had several times before with the driving analogy: when we drive, we know the consequences. When we measure temperature, we are sensing reality. But when a cipher fails we have no indication of failure.

When there is no indication of failure, there is nothing to extrapolate. And when there is no measure for the thing which fails, there is no meaning to extrapolation.

>That's basic physics, btw. :-)

And we see just how well it did.

>>Presumably, you would handwave about what our Opponents can do both
>>now and in the future and say that caution is silly. But that
>>conclusion is based on your opinion that we can predict what others
>>may do in the future, which I find very strange. If that were true in
>>general, we could put criminals in jail before they did anything.
>
>This author makes no distinction between being able to predict
>something with 100% accuracy and being able to predict something with
>lesser accuracy. For example, magician and card-sharp John Scarne once
>described playing gin rummy (for money) with a player who, after
>shuffling the cards, would square up the deck with the bottom facing
>towards him. An innocent-seeming idiosyncracy, except that he now knew
>the bottom card in the deck (which in gin rummy never comes into
>play). Suppose this card were the 8 of hearts; the player cannot
>predict, with 100% accuracy, what the next card in the deck will be,
>but he knows it will not be the 8 of hearts. This seemingly
>insignificant piece of information gives him a huge advantage; he
>knows that there is little percentage in trying to fill a meld of 8's
>or a run of 6-7-8 or 8-9-T of hearts, and none whatsoever in trying to
>fill an inside run of 7-8-9.
>
>In studying PRBGs, ability to predict the next bit with a probability
>of 0.5 + epsilon, where epsilon is a small number (usually on the
>order of 1/polynomial(log n)) can be a huge advantage.

I assume this analogy is intended to show that in some cases one can use past observations to usefully predict the future. Such is the role of most industrial knowledge. But this analogy is inappropriate for the issue being discussed.

The issue is whether cryptanalytic results can be used to compare the strength of ciphers with respect to the future abilities of unknown Opponents. In the above analogy, the Opponent is known, his weakness already judged, and ongoing results measurable. The cryptography issue has no such convenient touchstones.

>>With respect to the problem of potential catastrophic failure from a
>>single-cipher system, no amount of cryptanalysis can prevent such
>>failure. Both untested ciphers and massively-tested ciphers are the
>>same in the sense that neither can be trusted.
>
>Rubbish. I don't trust Charlie the counterfeiter, Ernie the embezzler,
>Rocco the rapist, or Sammy the serial killer. But I can give you a
>rough estimate of who I'd *least* like to crash my sister's slumber
>party.

I have no idea what this means.

I see no reason to change my statement, since it is correct as it
stands.

I suppose the issue here is your interpretation of "trust," which I
touched on earlier.

Terry Ritter ritter@io.com http://www.io.com/~ritter/
Crypto Glossary http://www.io.com/~ritter/GLOSSARY.HTM

Subject: Re: Question on confidence derived from cryptanalysis.
Date: Wed, 21 Apr 1999 23:58:27 GMT
From: jsavard@tenMAPSONeerf.edmonton.ab.ca (John Savard)
Message-ID: <371e6357.27881073@news.prosurfr.com>
References: <371e4437.15431777@news.io.com>
Newsgroups: sci.crypt
Lines: 27

ritter@io.com (Terry Ritter) wrote, in part:

>I guess "yes." If there is just one standard cipher, the issue is not
>so much what any one of us has to lose as it is what society as a
>whole has to lose.

>From this I conclude that the use of a single standard cipher
>throughout society would be an unthinkable risk.

Here, you and I are in agreement. New attacks are being found against
symmetric block ciphers, such as the boomerang attack and the slide attack.
Also, one of the papers on the NIST site is called "Future Resiliency", and
it is a defense of that point of view.

However, I don't think that for the AES process to pick one winner will
lead to that situation, any more than the existence of DES has stopped
people from using IDEA or Blowfish.

If anything, I'm more worried about a lot of messages suddenly becoming
readable through a catastrophic failure of public-key cryptography. But
such a failure at least is likely to become public knowledge; an
and secretly breaking "the" block cipher without anyone knowing certainly
is a real possibility.

John Savard (teneerf<-)
http://members.xoom.com/quadibloc/index.html

Subject: Re: Question on confidence derived from cryptanalysis.
Date: Fri, 23 Apr 1999 22:46:33 GMT
From: "Douglas A. Gwyn" <DAGwyn@null.net>
Message-ID: <3720F809.52DB949B@null.net>
References: <371b711c.6682075@news.calgary.telusplanet.net>
Newsgroups: sci.crypt
Lines: 12

Earth Wolf wrote:
> Remember Pearl Harbour, where warning of the impending attack
> arrived by messenger amid the smoking aftermath?

That's not what happened. The Japanese cleverly misled our
intelligence analysts into believing that their fleet was still
in home waters. What indication we had was that hostilities were
about to commence, *not* that Pearl would be the actual target.

That's the "executive summary" of the outcome of a massive
Congressional investigation, record of which are available in
the National Archives II (Modern Military History branch).

Subject: Re: Question on confidence derived from cryptanalysis.
Date: Mon, 19 Apr 1999 21:49:34 GMT
From: "Douglas A. Gwyn" <DAGwyn@null.net>
Message-ID: <371BC0D6.158C18E9@null.net>
References: <371B69DC.17EE09E1@raas.co.nz>
Newsgroups: sci.crypt
Lines: 20

Geoff Thorpe wrote:
> ... I dare say the NSA et al have less motivation to
> chase down some of the more daunting theoretical possibilities for

> weaknesses in algorithms, especially when in reality, so many of them
> lead nowhere or to advances that are at best - theoretical.

For example, several "significant" results in academic papers say
that certain systems can be broken with an inordinate amount of
resources, if 2^24 chosen plaintexts are used. It's hard to justify
such work when your job performance is measured by practical results
"in the field".

Generally speaking, Terry is right to be concerned over the unknown,
but some risks are greater than others. The specific algorithms you
mentioned previously are among the better risks. If the stakes are
really high, thoroughly studied systems are better bets than untested
ones. That's not to say that we don't need new, better systems, but
it takes *time* to subject them to enough testing and analysis to
develop confidence in them. Maybe some day we'll all understand that
Terry's approach (or David's) is a better way to go -- or maybe not.

Subject: Re: Question on confidence derived from cryptanalysis.
Date: 19 Apr 1999 23:21:54 GMT
From: David A Molnar <dmolnar@fas.harvard.edu>
Message-ID: <7fgdqi$k09$1@news.fas.harvard.edu>
References: <371BC0D6.158C18E9@null.net>
Newsgroups: sci.crypt
Lines: 62

Douglas A. Gwyn <DAGwyn@null.net> wrote:
> For example, several "significant" results in academic papers say
> that certain systems can be broken with an inordinate amount of
> resources, if 2^24 chosen plaintexts are used. It's hard to justify
> such work when your job performance is measured by practical results
> "in the field".

I agree with you about 85%. The other 15% comes from refinements of those
attacks which make them more practical, and the cases where bad design of
a system make them relevant. For example, it was known that knapsack
public-key systems leaked bits of information long before any specific
catastophic results were known. The single bit doesn't help much, but
acts as a warning sign that something is wrong. Now knapsacks are the
prime example of crypto that seemed 'pretty secure' and wasn't. i

Adaptive chosen ciphertext attack is a very strong attack, requiring that
the adversary decrypt values of its choice on your equipment, and perhaps
lots of them. It is not obvious how someone would apply it to a real
world system. Yet Daniel Bleichenbacher found that some implementations
of SSL aren't secure against it. Even though that attack is just barely
on the edge of practicality, we now have a new RSA PKCS standard.

Then you can improve attacks by gaining more information about what, exactly,
it is that you're attacking. There's a paper in Crypto '98 (for the life
of me I can't find it now, I'm sorry) on "From Differential
Cryptanalysis To Ciphertext Only Attacks." It uses the assumption that
the cryptanalyst is dealing with English text to turn chosen-plaintext
attacks into ciphertext-only attacks. I can't find it, or else I'd report
how efficient the new attacks are -- but this is a qualitative difference
in utility. It wouldn't be possible without the earlier work.

My point is that there's enough precedent that I can imagine a boss
with foresight not being too dismayed by the "2^24 chosen plaintexts,
needs 2^42 operations and 2^56 blocks of memory" sort of result.
I can imagine that she wouldn't be thrilled, but I can also imagine
that she'd try to follow it up, too.

> Generally speaking, Terry is right to be concerned over the unknown,
> but some risks are greater than others. The specific algorithms you
> mentioned previously are among the better risks. If the stakes are
> really high, thoroughly studied systems are better bets than untested
> ones. That's not to say that we don't need new, better systems, but
> it takes *time* to subject them to enough testing and analysis to
> develop confidence in them. Maybe some day we'll all understand that
> Terry's approach (or David's) is a better way to go -- or maybe not.

Thank you for referring to it that way, but I'm rather new to the approach.
I suspect my enthusiasm comes from its novelty, as well as the prospect
of finally being able to "measure" security. :-)

So far provable security doesn't seem to do much for block ciphers, though...
at least that I've seen (and I've heard about DFC but haven't looked at it
much yet), or indeed quick bulk ciphers of any kind. That leaves
Terry's approach and whatever you want to call the other.

Honestly, I need to read more about ciphers by ritter and see what this
'scaled down to experimental size' means, along with everything else.

Thanks,
-David Molnar

Subject: Re: Question on confidence derived from cryptanalysis.
Date: Tue, 20 Apr 1999 06:37:03 -0400
From: "Trevor Jackson, III" <fullmoon@aspi.net>
Message-ID: <371C58CF.286794F7@aspi.net>

References: <371B69DC.17EE09E1@raas.co.nz>
Newsgroups: sci.crypt
Lines: 205

Geoff Thorpe wrote:
>
> Hello,
>
> "Douglas A. Gwyn" wrote:
> > (I'm not suggesting that academics haven't made useful
> > contributions to the state of the art, just that their
> > work does not define the total state of the art.)
>
> Oh I agree completely - I was just taking issue with what I perceived to
> be the following idea: Until we actually break it, or *prove* it secure,
> we have no more measure of strength for it than for another (less
> "investigated") one. I feel that quite the opposite is true - it IS a
> very appropriate statistical measure of strength, and moreover the only
> realistic one we have to work with. If the total state of the art stays
> roughly in sync with the academics, albeit "they" may have a couple of
> things up their sleeves and they may often get the jump on us by a few
> months/years with various developments, then we can make reasoned
> guestimations on the strength of a cipher against them based on the
> strength of a cipher against us.

This usage of the term strength may be inappropriate. As a substitute I offer the term confidence. The difference is partly connotative, but an example may illustrate a real distinction worth preserving. In a restricted set of cases one might use a weak cipher, knowing that it is theoretically breakable, but also knowing that the adversaries (threat model) cannot break it in practice. The reasons for their inability might be lack of resources, cruptographic sophistication, or as simple as one message sent one time. In this situation we have confidence that the cipher will protect the secret, but the cipher is not strong.

The practical test of ciphers is valid in the sense that it can give us confidence, but it cannot give us strength. In a sense the practical test of exposure to many and varied attacks gives us a kind of lower bound on the types of attacks a cipher might not resist. Once a cipher has survived a gauntlet, we know that any successful attack must be fairly sophisticated, optomized agains the particular cipher, or simple but based on a radical insight or advance in the field (linear, differential, boomerang, or sliding attacks appear to be advances).

But this lower bound does not tell us *anything* about the "strength" of the cipher (the units for which are completely undefined), but tells us a lot about the confidence we might repose in the cipher.

Ritter appears to be after strength. Thorpe appears to be after confidence.

>
> > > Mathematical problems - that is what I was referring to ...
> >
> > Yes, cryptology is largely applied mathematics, but
> > practical cryptanalysis has evolved in the context of
> > operational experience that is largely unavailable to
> > outsiders, and that has caused a substantial difference
> > between insiders and outsiders in their aims and methods.
>
> Well yes and no ... applied mathematics has a handy of way of pushing
> things along nicely particularly in the area of computation (complexity)
> problems. "Their" ideals may be different to ours but I doubt their aims
> or methods are ... everybody would love to break an established cipher
> (with the possible exception of the patent holder), everybody would love
> to *prove* a cipher secure (with the possible exception of the patent
> holder's competition). I dare say the NSA et al have less motivation to
> chase down some of the more daunting theoretical possibilities for
> weaknesses in algorithms, especially when in reality, so many of them
> lead nowhere or to advances that are at best - theoretical.
>
> "They" have budgets (albeit big ones) and they probably have things
> they'd rather spend it on (satelites, lobbying, hardware, breaking
> implementations, breaking installations, etc). OTOH, having been
> post-grad in a mathematics department before I know very well that this
> obsession for looking in every nook and cranny of all things theoretical
> is exactly the sort of thing academics get off on. Cracking ciphers (ie.
> the actual algorithm itself, not the practical implementation details
> that may be vulnerable) is much more the meat and veg of academics who
> like to play and write papers. "They" just want information, and I'm
> guessing just do whatever they got to do to get it - and searching
> endlessly for little theoretical weaknesses is probably not their top
> priority. That's not to say they don't do it and do it very well, but I
> doubt their considerable advantages in resources are put so much to this
> task as to make our abilities so incomparable or unrelated as some might
> believe.

A good point. However, we canot deal with their (secret) intentions, but must anticipate their possible (even more secret) capabilities. Thus amplifying the threat model is a sensible thing to do. It eliminates some of the risk of catastrophically underestimating them by enhancing the risk of expensively overestimating them.

An appropriate paranoia dictates that we accept the costs of overestimation.

>
> > Some problems, like efficient factoring, are obviously
> > relevant, and unlikely to be achieved in secret without
> > happening in the outside around the same time. Other
>
> I agree but I doubt very much Mr Ritter does.
>
> > breakthroughs have been kept secret for decades, in some
> > cases. So there really is reason to fear that the most
> > advanced "enemies" might know how to easily crack some
> > system you use that appears uncrackable to all outsiders.
>
> I know - and there's a lot of targets out there so the odds are on that
> at least one of them has fallen completely to an "unpublished" source
> without our knowing it. However, I just think it's more likely to be
> something less well analysed in the "open" than something well analysed
> in the "open" for the reasons I've mentioned, and that Mr Ritter doesn't
> agree with.

It appears to me that he *does* agree (tho he can certainly speak for
himself), which is why he has repeatedly proposed the use of multiple
ciphers both to spread eggs across baskets, and to provide layered
security where warranted.

>
> On a related note (and all IMHO), bit-twiddling little ciphers are no
> less "mathematical" than effecient factoring. Discrete maths actually
> finds that cute little "permutation stuff" quite fashionable from my
> limited contact with it (and them). Factoring tends to interest (again
> from my limited contact) more the applied heads - meaning I'd give
> better odds to developments in faster optimizations on 64-bit platforms
> with super-dooper cache, than to fundamental breaks in the factoring
> algorithms [;-)
>
> > There is a significant difference between what is "well
> > established" in long-existing, well-funded cryptologic
> > organizations and what is "well established" in the
> > dispersed, high-turnover-rate academic community.
>
> True - and to agree with Mr Ritter for a moment, I think perhaps another
> risk is that the academics tend to show interest in things that interest
> them - where as the well-funded organisations you speak of more likely
> show interest in things that give them the best chance of accomplishing
> some objective. However, this pragmatic mind-set, albeit fearsome in
> many ways, might give us some hope that in the ethereal hights of trying
> (and hoping) to break an already well-studied algorithm, they probably
> are less hopeful, less obsessed, and more practical and realistic. After
> all, RSA, IDEA may be perfect but if Win95's TCP allows a
> password-sniffer to leak into your PC "they" have accomplished their
> objective and "broken" PGP as far as "they" are concerned.
>
> > There is a big problem in working in *applied* fields
> > academically, since it is harder to get academic
> > respect from publication of application or tutorial
> > papers instead of research papers. There are many
> > technologies that are well-known *in general* in the
> > research community, but their specific application to
> > cryptology is *not* well known.
>
> Probably quite true.
>
> > > We all know that risk - it's the probabilities that are open to
> > > debate. ...
> >
> > More precisely, the likelihoods.
> > The nice thing is that *relative* likelihoods can be estimated
> > and used to make decisions; e.g. "I need a cipher that <meets
> > certain requirements> -- pick one."
> > If the consequences of not making a decision are sufficiently
> > severe, then even an uncertain decision can be better than
> > letting the uncertainty stop you from making a decision.
>
> Exactly, and well said.
>
> > > Me, I'm going to stick with RSA and triple-DES for a while.
> >
> > In a well-designed cryptosystem, these do seem sufficiently
> > secure against realistic threats for the near future. Any
> > vulnerabilities would most likely occur elsewhere in the
> > system/protocols, not in these encryption algorithms as such
> > (assuming of course a long RSA key, and 168-bit 3DES key).
>
> I think that too, but as Mr Ritter might say - you are already in the
> abyss and are naive if you think that. If that is so, I am comfortable
> in my naivety.
>
> > That seems to be part of Ritter's aim, but others seem to
> > think that during cryptanalysis the stages have to be peeled
> > like an onion, and they assume that there is not enough
> > pattern available at the next-to-outermost layer for there
> > to be any chance of peeling the outer layer off.
>
> Well hopefully someone will look at this, and demonstrate some success
> from it. Results speak for themselves, even to ivory tower academics

> [;-)
>
> > > And this can't be achieved within ONE cipher? When you start talking
> > > multiple algorithms, you instantly start talking interoperability
> > > and standardisation headaches.
> >
> > That's a significant concern, because breakdowns in operational
> > procedure often provide the enemy analyst the entering wedge he
> > needs to crack a system.
>
> Exactly, and if I resort to using a different cipher every week ... the
> cryptanalysts will not keep up with them satisfactorily and I have a lot
> more confidence that "they" WILL be breaking my traffic on a
> semi-regular basis.

Layered algorithms do not dictate expensive or complex operational
requirements. The implementation of a layered cipher needs some care,
but no more than any other secure system. This issue appears to be a
red herring.

Subject: Re: Question on confidence derived from cryptanalysis.
Date: Tue, 20 Apr 1999 00:28:14 -0400
From: Geoff Thorpe <geoff@raas.co.nz>
Message-ID: <371C025E.6AD4BAB8@raas.co.nz>
References: <371C58CF.286794F7@aspi.net>
Newsgroups: sci.crypt
Lines: 131

Hello,

"Trevor Jackson, III" wrote:
> This usage of the term strength may be inappropriate. As a substitute I
> offer the term confidence. The difference is partly connotative, but an
> example may illustrate a real distinction worth preserving. In a

Well originally this grew out of the idea of "tested strength" (whatever
that might mean). I dropped the "tested" after a while [;-) By the way,
it was me who chose to rename the subject of this branch of the thread
... so I do agree with you.

> The practical test of ciphers is valid in the sense that it can give us
> confidence, but it cannot give us strength. In a sense the practical
> test of exposure to many and varied attacks gives us a kind of lower
> bound on the types of attacks a cipher might not resist. Once a cipher
> has survived a gauntlet, we know that any successful attack must be
> fairly sophisticated, optomized agains the particular cipher, or simple
> but based on a radical insight or advance in the field (linear,
> differential, boomerang, or sliding attacks appear to be advances).

I totally agree.

> But this lower bound does not tell us *anything* about the "strength" of
> the cipher (the units for which are completely undefined), but tells us
> a lot about the confidence we might repose in the cipher.

Well it gives us a sort of "measure" that we can apply when we make our
choice ... something new, has held out for just a little while, but
really motors along on a RISC chip - might lead to one choice. Something
old, battled, and stubborn - might be more appropriate where performance
is less important and it's wiser to be ultra-conservative about
security.

> Ritter appears to be after strength. Thorpe appears to be after
> confidence.

We're probably all after both.

As this discussion keeps dividing in cellular fashion, I can't really
reply to each post that was a reply to one of mine - so, in another
"branch" Terry said the following;

Terry Ritter said:
> In summary: 1) We cannot estimate the probability that an effective
> attack exists which we did not find; and 2) We cannot estimate the
> probability that even if such an attack does exist, our Opponents can
> find it and use it. I thus claim that we CAN know nothing of the
> probability of future cipher failure, and cannot even reason that this
> probability is "small." The practical consequence of this is that we
> cannot trust any cipher.

I disagree - and I disagree with every sentence moreover. I may not
design ciphers but I can definately slug it out with most people
regarding probability theory, statistics, and logic. I also have to
assist with various API designs and have been on the (l)using end of
quite a few if we want to talk standards, picking algorithms, and
covering butts (oh yeah, I've done quite a bit of Risk Management
related stuff too).

Now, statement (1) is wrong. Maybe you cannot make estimates, and maybe
you do not like the estimates others may employ. But there are ways to
make estimates whose rationalisation is acceptable to those involved.
That includes me. You also referred in that post to a complete lack of
evidence but I think you yourself would be well positioned to refute

that. Take every damned cipher you ever heard of (with any degree of cryptanalysis against it), exercise some personal judgement as to some measure of time+effort that the cipher was subjected to (by publishing authors - obviously not the spooks) before it became widely regarded as unacceptable, and take a look at the resulting distribution. That may not be a precise science, and of course it involves warm-fuzzy personal interpretations (time+effort) but it is not unacceptable for many people, particularly those who would otherwise be rendered with NO effective way to evaluate. I dare say that your distribution, if you've made semi-reasonable interpretations along the way, will show that a ciphers that lasted 10 years had a much better chance of lasting another year than the average "expected life". It's a very basic and common mathematical model/argument, and it's common sense.

I've already explained why I think that (2) is wrong - nobody knows any of this stuff FOR SURE, but you make a call when you don't have perfect information. Our Opponents are just well-paid versions of us, most of whom probably grew up around us, and who find their occupations not too unfathomably unethical to suffer day by day. I still maintain that what we can do and achieve is a statistical, probabilistic, and "confidence" variable that does not run along independantly of theirs. Depends how much George Orwell you read though ...

> > like to play and write papers. "They" just want information, and I'm
> > guessing just do whatever they got to do to get it - and searching
> > endlessly for little theoretical weaknesses is probably not their top
> > priority. That's not to say they don't do it and do it very well, but I
> > doubt their considerable advantages in resources are put so much to this
> > task as to make our abilities so incomparable or unrelated as some might
> > believe.
>
> A good point. However, we canot deal with their (secret) intentions,
> but must anticipate their possible (even more secret) capabilities.
> Thus amplifying the threat model is a sensible thing to do. It
> eliminates some of the risk of catastrophically underestimating them by
> enhancing the risk of expensively overestimating them.

Sure thing - but the whole system does not collapse down to a binary system of "broken" and "not-broken-yet" ... as you say, you put together a threat model ... consistent with your requirements and using a chosen method for judging a components "worth", and amplify it here and there as appropriate. A lot like putting together a cost-proposal I guess ... add in your known prices, choose an acceptable value for the "unknowns", amplify the costs of all the "risky" bits, add x% profit on top - and then bang another 30% on top for good measure, and generally covering your butt some more.

> It appears to me that he *does* agree (tho he can certainly speak for
> himself), which is why he has repeatedly proposed the use of multiple
> ciphers both to spread eggs across baskets, and to provide layered
> security where warranted.

3 ciphers strung in a line is, to me, a cipher. You need all three in the same place and in the same order to have anything other than a "noise generator". Breaking 3 ciphers should be no more difficult than breaking one well designed one using 3 different stages (if a cipher is based on one "idea", "primitive", or whatever then your vulnerability must surely be higher than distinct ideas employed serially?). It seems the argument put forth was more one of splitting the traffic (conceptually across time and application, not packet by packet I assume) across ciphers, and rotating the old out and the new in on a regular basis. I see this as unacceptable in a real-world scenario for reasons of interoperability & standardisation, as well as security.

Cheers,
Geoff

Subject: Re: Question on confidence derived from cryptanalysis.
Date: Tue, 20 Apr 1999 05:52:19 GMT
From: ritter@io.com (Terry Ritter)
Message-ID: <371c15e3.8290372@news.io.com>
References: <371C025E.6AD4BAB8@raas.co.nz>
Newsgroups: sci.crypt
Lines: 176

On Tue, 20 Apr 1999 00:28:14 -0400, in <371C025E.6AD4BAB8@raas.co.nz>, in sci.crypt Geoff Thorpe <geoff@raas.co.nz> wrote:

>[...]
>Terry Ritter said:
>> In summary: 1) We cannot estimate the probability that an effective
>> attack exists which we did not find; and 2) We cannot estimate the
>> probability that even if such an attack does exist, our Opponents can
>> find it and use it. I thus claim that we CAN know nothing of the
>> probability of future cipher failure, and cannot even reason that this
>> probability is "small." The practical consequence of this is that we
>> cannot trust any cipher.
>
>I disagree - and I disagree with every sentence moreover. I may not
>design ciphers but I can definately slug it out with most people
>regarding probability theory, statistics, and logic.

You may be willing to "duke it out," as though this were some sort of winner-take-all contest, but if you believe your logic is compelling, you will have to think again. Not only am I not compelled, I am appalled to see you repeating things over and over, in the apparent illusion that this has some relation to logic or scientific argument.

>I also have to
>assist with various API designs and have been on the (l)using end of
>quite a few if we want to talk standards, picking algorithms, and
>covering butts (oh yeah, I've done quite a bit of Risk Management
>related stuff too).

What a guy you are I'm sure. Let's get on with it:

Recall that my position does not rest upon an estimation of someone else's capabilities. It is not my *opinion* that any cipher we have *might* possibly break -- that is fact. I assume the worst case, and propose systems to provide strength even then.

Your position, dare I state it, is that you *can* estimate the capabilities of your Opponents. You also say you can estimate the future strength of a cipher from past tests. But for all this claiming, we see no similar statements in the scientific literature. So these are simply your opinions, and I see no supporting facts.

>Now, statement (1) is wrong.

Which was: "1) We cannot estimate the probability that an effective attack exists which we did not find."

Since you think this is wrong, you must believe we *can* make an estimate. Fine. Do it. Show me.

>Maybe you cannot make estimates, and maybe
>you do not like the estimates others may employ. But there are ways to
>make estimates whose rationalisation is acceptable to those involved.
>That includes me.

Alas, what people believe is not science.

>You also referred in that post to a complete lack of
>evidence but I think you yourself would be well positioned to refute
>that. Take every damned cipher you ever heard of (with any degree of
>cryptanalysis against it), exercise some personal judgement as to some
>measure of time+effort that the cipher was subjected to (by publishing
>authors - obviously not the spooks) before it became widely regarded as
>unacceptable, and take a look at the resulting distribution. That may
>not be a precise science, and of course it involves warm-fuzzy personal
>interpretations (time+effort) but it is not unacceptable for many
>people, particularly those who would otherwise be rendered with NO
>effective way to evaluate. I dare say that your distribution, if you've
>made semi-reasonable interpretations along the way, will show that a
>ciphers that lasted 10 years had a much better chance of lasting another
>year than the average "expected life". It's a very basic and common
>mathematical model/argument, and it's common sense.

Oddly, no such study has appeared in the literature. That seems somewhat strange, since you say it is very basic common sense. Perhaps everyone else in cryptography has simply been blinded to this fundamental truth. When will you write it up for us?

You are arguing your opinion about cipher strength. (Recall that I do not argue an *opinion* about cipher strength, but instead the *fact* that any cipher may be weak.) If you have compelling factual evidence, I will support it. Show me the correlation you say exists. Prove it. Then you can use it.

>I've already explained why I think that (2) is wrong - nobody knows any
>of this stuff FOR SURE, but you make a call when you don't have perfect
>information.

Nobody has any problem with you making a call for yourself and risking only yourself. But if this "call" is intended to formulate what "should" happen for much of society, you may need to revise your estimate as to the consequences of failure. Just how much disaster are you willing for us to have?

Will it be OK for everyone to use the single standard cipher which you predict is strong, if you turn out to be wrong? Will it be OK when communications grind to a halt and incompatible low-security temporary measures are instituted everywhere while a new cipher is integrated into all the programs which must be replaced throughout society? Is that OK with you?

>Our Opponents are just well-paid versions of us, most of
>whom probably grew up around us, and who find their occupations not too
>unfathomably unethical to suffer day by day.

This is simply breathtaking: It is appallingly, plainly, immediately false even to the most casual observer. People do differ, we do have

limitations others do not have, and others often do take advantage of knowledge to which we have no access.

>[...]
>Sure thing - but the whole system does not collapse down to a binary
>system of "broken" and "not-broken-yet" ... as you say, you put together
>a threat model ... consistent with your requirements and using a chosen
>method for judging a components "worth", and amplify it here and there
>as appropriate. A lot like putting together a cost-proposal I guess ...
>add in your known prices, choose an acceptable value for the "unknowns",
>amplify the costs of all the "risky" bits, add x% profit on top - and
>then bang another 30% on top for good measure, and generally covering
>your butt some more.

Write whatever numbers you want: you cannot support them.

>> It appears to me that he *does* agree (tho he can certainly speak for
>> himself), which is why he has repeatedly proposed the use of multiple
>> ciphers both to spread eggs across baskets, and to provide layered
>> security where warranted.
>
>3 ciphers strung in a line is, to me, a cipher.

The distinction is that each cipher is an independent and separable element which can be "mixed and matched" with any other. Each cipher is tested as an independent unit, and brings whatever strength it has independent of internal ciphering requirements. Dynamic mixing and matching prevents having any fixed target to attack.

>You need all three in
>the same place and in the same order to have anything other than a
>"noise generator". Breaking 3 ciphers should be no more difficult than
>breaking one well designed one using 3 different stages

Really? Tell me more about how ciphers are designed. How long did you say you have been doing this? How many ciphers have you designed? Have you measured them? Where can we see your work?

>(if a cipher is
>based on one "idea", "primitive", or whatever then your vulnerability
>must surely be higher than distinct ideas employed serially?). It seems
>the argument put forth was more one of splitting the traffic
>(conceptually across time and application, not packet by packet I
>assume) across ciphers, and rotating the old out and the new in on a
>regular basis. I see this as unacceptable in a real-world scenario for
>reasons of interoperability & standardisation, as well as security.

What you mean is that *you* do not know how such a system would be standardized and made interoperable. Then you imply this means *nobody* *else* could do it either.

I note that this is precisely how you reason about the cryptanalytic capabilities of others: It is false here, there, and everywhere you present it. You are not the best example, not everyone is like you, and it is invalid to assume others have your limitations.

Terry Ritter ritter@io.com http://www.io.com/~ritter/
Crypto Glossary http://www.io.com/~ritter/GLOSSARY.HTM

Subject: Re: Question on confidence derived from cryptanalysis.
Date: Tue, 20 Apr 1999 20:35:53 -0400
From: Geoff Thorpe <geoff@raas.co.nz>
Message-ID: <371D1D69.AF9907B6@raas.co.nz>
References: <371c15e3.8290372@news.io.com>
Newsgroups: sci.crypt
Lines: 347

Hi,

Terry Ritter wrote:
>
> On Tue, 20 Apr 1999 00:28:14 -0400, in <371C025E.6AD4BAB8@raas.co.nz>,
> in sci.crypt Geoff Thorpe <geoff@raas.co.nz> wrote:
> >
> >I disagree - and I disagree with every sentence moreover. I may not
> >design ciphers but I can definately slug it out with most people
> >regarding probability theory, statistics, and logic.
>
> You may be willing to "duke it out," as though this were some sort of
> winner-take-all contest, but if you believe your logic is compelling,
> you will have to think again. Not only am I not compelled, I am
> appalled to see you repeating things over and over, in the apparent
> illusion that this has some relation to logic or scientific argument.

Other parts of your posts refer to your ideas and your technologies and your experience, etc. I do not claim familiarity with your ideas, but moreover I was attempting to say that I also do not claim to be a cipher designer. I am a scientist however, and was tiring of your attempts to state what I saw as broad, arrogant, and overly pessimistic views as

fact together with implications of naivety and ignorance on my (and others?) part. I also have no desire to "duke it out", "lock horns", or any such thing - just wanted to make sure you understood that not being a cipher designer does not mean I'm going to lie down, take your statements as authorative when I genuinely disagree with some of your fundamental points.

> >I also have to
> >assist with various API designs and have been on the (l)using end of
> >quite a few if we want to talk standards, picking algorithms, and
> >covering butts (oh yeah, I've done quite a bit of Risk Management
> >related stuff too).
>
> What a guy you are I'm sure. Let's get on with it:

yadayadayada. I have a vague idea now of some of your areas of expertise as per your posts and the peripheral discussion. You seem to have no tolerance for my views on the matter so I thought it appropriate to at least let you know that I'm not some bunny out on a limb here. However, I'm of the impression that my problem here is not that you won't consider my opinion as worthy of some merit, so much as you won't consider any other opinion than your own as worthy of merit. Mind you, I recall that recently you categorically discarded the considered views of Mr Schneier and others so I guess credentials are a waste of time anyway - I should have thought of that.

> else's capabilities. It is not my *opinion* that any cipher we have
> *might* possibly break -- that is fact. I assume the worst case, and
> propose systems to provide strength even then.

Exactly, you assume the worst case. Whilst you certainly will never be accused of lacking precaution, why should I accept that your position the only appropriate one to adopt? The world operates a lot more pragmatically than you might be prepared to accept, and naturally we roll the dice as a result - memories of the ice-storm in Montreal and the massive power-outage in Auckland, New Zealand (particularly relevant to me) flood to me at this point. Individually, each failure is roundly criticised and everyone pats themselves on the back as to why they wouldn't have fallen into that particular trap.

I could get killed the very next time I go driving, in fact I'm increasingly of the opinion there are those who wouldn't be overly upset about it. But I do not insist that I and others must push through radical measures involving gondolas, pulleys, and the abolition of personal automotive ownership.

Before I get accused of doing precisely what I don't want to do (lock horns, duke it out, etc) ... let me just say that I really am warming to an idea implicit in all of this - and I believe it is one of yours, though it was Trevor I think who recently illustrated it quite well ... namely the employment of a standard bank of ciphers that can be invoked on demand in any number of possible configurations eg strung together in a different order every time, utilising the different modes of operation, etc etc. I also agree the implementation and standardisation headaches of this sort of scheme are not insurmountable - indeed every standard SSL implementation I've seen lately seems to implement most of the core ciphers/modes that could be used in such a scheme. I'm also definately not against the idea of extensibility of frameworks to incorporate as-yet-unknown elements - indeed PKCS#7 probably didn't have DSA, ElGamal etc in mind, but now they seem to be creeping into CMS and that method seems to allow room to grow it again later. (If I've confused this with something else, someone please correct me - I could have my wires a little crossed right now and don't have any reference handy).

But it seems to me, especially with regard to non-realtime applications, that to an extent, less-is-more ... sure a few ciphers in your pool is fine, especially if everyone has them. But the wholesale liberalisation of
cipher farming seems to create a very real problem - a kind of protocol grid-lock. And frankly, I still place a lot of stock in what *I* rank as ciphers of tested strength and wouldn't want any system of mine having too many "new toy" ciphers creeping in. Perhaps we need to agree to disagree.

> Your position, dare I state it, is that you *can* estimate the
> capabilities of your Opponents. You also say you can estimate the
> future strength of a cipher from past tests. But for all this
> claiming, we see no similar statements in the scientific literature.
> So these are simply your opinions, and I see no supporting facts.

Scientific literature? Ironic that it is precisely this quantity that you appear to place very little value in with regard to ("tested") cipher strength, and yet I am supposed to find some to support my view? Anyway - I have already said that my view (that a cipher not falling over despite some considerable efforts against it does merit some "value") is not based on any exact science. I think history, and some basic common sense warrant my conclusions. Your contrary opinion does not appear to be any more scientifically founded - although it does appear to be a little more "absolute" or "axiomatic" (and IMHO "not terribly practically useful").

> >Now, statement (1) is wrong.
>
> Which was: "1) We cannot estimate the probability that an effective

> attack exists which we did not find."
>
> Since you think this is wrong, you must believe we *can* make an
> estimate. Fine. Do it. Show me.

The fact that I can drive in Quebec without getting killed for 3 months suggets I can probably survive another few days. I don't know what my chances would be in London - and maybe the insurance salesman doesn't either. Fine, I'll go for a drive shortly and if I STILL don't get killed (ie. I post again in the future to this group) then that supports my estimate of the probability. If you think I'm wrong, break triple-DES and you show me. Otherwise - neither of us is correct in any pure sense ... but I'm still comfortable with my approach and if others are too that's all that matters. Anyway, now I think about it further - exactly how can you possibly insist that "we cannot estimate a probability" ??? Sounds absurd. Particularly with something that has any historical record at all?

As someone with a love of pure mathematics, it does feel a little disturbing to be arguing a point with someone where it is *I* who am on the fuzzy, pragmatic, approximation side of the fence and the other is arguing puristically.

> Alas, what people believe is not science.

But what people believe influences what they will and will not do (and will or will not put up with). And unless a scientist can *prove* absolutes they will have difficulties imposing absolutes. Perhaps a good way to measure this is to ask an insurance-brokerage expert to comment on the insurability (premiums etc) on an information resource secured using your approach versus something like I prefer. Not a single ounce of "science" will enter into this equation (I suppose) and yet I can't imagine a more adequate way to judge the situation - after all, it is these kind of people whose lives it is to cover the costs of things when they go wrong.

> >year than the average "expected life". It's a very basic and common
> >mathematical model/argument, and it's common sense.
>
> Oddly, no such study has appeared in the literature. That seems
> somewhat strange, since you say it is very basic common sense.
> Perhaps everyone else in cryptography has simply been blinded to this
> fundamental truth. When will you write it up for us?

If I hire a programmer to work with a new technology and a deadline, and my options (for the same money/conditions etc) are between someone who has demonstrated he/she can handle new technologies (in the past of course), and someone who *might* be able to handle new technologies, I'm going to hire the one with experience. A new candidate might be faster, hungrier, and actually better with the new technology - but why would I take that chance versus the chance the experienced one ran out of puff? True, until I try one I will not know which one was better but I'll hope you agree estimations, probabilities, and common sense are all present and can be utilised. I got a feel that your view on this was almost quantum mechanical - then I remembered that even QM admits probability result and an unlikely one even though each is possible until you find out for sure).

But I digress perhaps, and we've already demonstrated we don't agree here so ...

> You are arguing your opinion about cipher strength. (Recall that I do
> not argue an *opinion* about cipher strength, but instead the *fact*
> that any cipher may be weak.) If you have compelling factual
> evidence, I will support it. Show me the correlation you say exists.
> Prove it. Then you can use it.

I've already admitted that my "correlation" is a fuzzy one, full of ideas that are "compelling" (to me) here, "suggestive" (to me) there, etc - and that my conclusion is a fuzzy one. Perhaps then I've shown compelling "fuzzy" evidence. [;-) Anyway, you are saying I cannot use "tested strength" as a measure - and your sole reason seems to be - "because it could still break tomorrow". Nobody disputes the latter statement but it does not logically imply the blanket assertion you make. Not proving things one way or the other does not mean we need default to your assertion, that all ciphers are equal when only existing failures to break them are in evidence, and abandon my assertion, that failing to break ciphers does provide useful information for "estimations".

And in case you ask, no - I know of NO research paper to support this and have no interest in attempting to create some when I'm already satisfied.

> Nobody has any problem with you making a call for yourself and risking
> only yourself. But if this "call" is intended to formulate what
> "should" happen for much of society, you may need to revise your
> estimate as to the consequences of failure. Just how much disaster
> are you willing for us to have?

The *consequences* of failure are not what I'm estimating. And again, I'll agree that the idea discussed before (utilising a defined set - for interoperability this seems necessary - of ciphers, algorithms, etc etc that can be jumbled around on the fly to diffuse the impact "a break" would have). It would interesting, though off topic, to see how your

absolutist approach generalises to arms control, transportation legislation, etc. All areas where "pragmatic fuzzies" tend to preside over "puristic absolutes" - even when they're cautionary variety.

> Will it be OK for everyone to use the single standard cipher which you
> predict is strong, if you turn out to be wrong? Will it be OK when

I've already moved a bit to your side on at least one point - one single cipher (if they are implicitly atomic and cannot encompass the idea that one can effectively put what would be 3 or 4 atomic ciphers into a "cipher") would not be as comforting as a small (I still think "fixed", or at least "slow moving") collection of ciphers jumbled up to disperse the impact a break in any one configuration would have. I still think my point applies to the selection of those ciphers though.

> communications grind to a halt and incompatible low-security temporary
> measures are instituted everywhere while a new cipher is integrated
> into all the programs which must be replaced throughout society? Is
> that OK with you?

And quantum computers could break everything and that wouldn't be OK with me either. But I'm not going to resort to carrier pigeons (which could be broken by a large society of hunters ... oh god ... this is getting too much).

> >Our Opponents are just well-paid versions of us, most of
> >whom probably grew up around us, and who find their occupations not too
> >unfathomably unethical to suffer day by day.
>
> This is simply breathtaking: It is appallingly, plainly, immediately
> false even to the most casual observer. People do differ, we do have
> limitations others do not have, and others often do take advantage of
> knowledge to which we have no access.

You still don't get what I'm saying ... YES people do differ, but I think continuously, not by quantum leaps that erase any relationship you can draw.

> >Sure thing - but the whole system does not collapse down to a binary
> >system of "broken" and "not-broken-yet" ... as you say, you put together
> >a threat model ... consistent with your requirements and using a chosen
> >method for judging a components "worth", and amplify it here and there
> >as appropriate. A lot like putting together a cost-proposal I guess ...
> >add in your known prices, choose an acceptable value for the "unknowns",
> >amplify the costs of all the "risky" bits, add x% profit on top - and
> >then bang another 30% on top for good measure, and generally covering
> >your butt some more.
>
> Write whatever numbers you want: you cannot support them.

You can be as cautious as you like and you could still get busted - you can be as irresponsible as you like and you COULD (not likely) get away with it. You can also just give up. That same model applies every time I write a proposal, an electricity company designs and insures an infrastructure, and many other real world situations. Tell me why I HAVE to resort to such a binary system of "broken" and "not-broken-yet". You don't seem to be able to support your claim that the test of time (and attack) does not provide a usable measure and you yourself have not written any numbers to try. Don't tell me and many other people with an interest that it's invalid to use such approaches, and then only support your claim by statement - particularly if you intend to then insist I support my own claims with numbers or proofs I'm supposed to pluck out of thin-air.

> >3 ciphers strung in a line is, to me, a cipher.
>
> The distinction is that each cipher is an independent and separable
> element which can be "mixed and matched" with any other. Each cipher
> is tested as an independent unit, and brings whatever strength it has
> independent of internal ciphering requirements. Dynamic mixing and
> matching prevents having any fixed target to attack.

So should good cipher design as far as I can see but I'll go along with you here. I see this idea as promising and will not argue with the premise that if you've got 5 good ones, why just stick with one - indeed why just stick with a fixed arrangement of that 5 (effectively making one very complicated, but still fixed, cipher) when you can jumble the order, modes of operation, etc each time. (The way in which that has been done would presumably become part of the "key"). I'd still prefer that we standardise on those 5 and maybe rotate new ones in "occasionally" (conservatively) in a method not-unlike the current AES process - ie. public exposure to candidates for a good hard thrash at them before actual incorporation of them into systems.

> >You need all three in
> >the same place and in the same order to have anything other than a
> >"noise generator". Breaking 3 ciphers should be no more difficult than
> >breaking one well designed one using 3 different stages
>
> Really? Tell me more about how ciphers are designed. How long did
> you say you have been doing this? How many ciphers have you designed?
> Have you measured them? Where can we see your work?

Already told you I'm not a cipher designer. But there are cipher designers who share my view so attack the idea, not the guy saying it. I

might also add - you're asking me to measure ciphers after having insisted quite strongly that any such attempt is implicitly impossible (with the absolute exception of breaking it).

Can I take apart a modern cipher and say "that's not a cipher - look, it's lots of little ciphers"? All I said was the division for me between 3 ciphers strung in a line and one cipher with 3 stages to it seems to be a question of packaging and patents. One could even stretch the definition and include the possibility of reording "stages" based on the "key". But I'm not going to suck myself into a bits-on-the-wire cipher designing discussion because I know I can't make a worthwhile contribution to it.

> >regular basis. I see this as unacceptable in a real-world scenario for
> >reasons of interoperability & standardisation, as well as security.
>
> What you mean is that *you* do not know how such a system would be
> standardized and made interoperable. Then you imply this means
> *nobody* *else* could do it either.

Fair call. Let me try again then - I think there could well be some very useful gains made from employing standards that use multiple primitives that hopefully seem (barring quantum computers?) to be independant targets of attack, that when used in different orders, modes, etc reduce the chances of the whole system getting broken rather than one putting the house on one primitive, or one configuration of the primatives. I do however think we should be measuring the primitives (which you suggest is pointless) as best we can, and that we should use a conservative approach to the standardisation on those primitives and the method by which new ones are incorporated into the standards.

If your "boolean model" of cipher strength is valid - can't this entire idea, when wrapped and considered as an entity in itself, then be implicated as just as "trust-worthy" as a single cipher that hasn't been broken? I would NOT regard them as equal but your argument, by extension, does.

Cheers.
Geoff

Subject: Re: Question on confidence derived from cryptanalysis.
Date: Wed, 21 Apr 1999 07:28:21 GMT
From: ritter@io.com (Terry Ritter)
Message-ID: <371d7e0a.11538318@news.io.com>
References: <371D1D69.AF9907B6@raas.co.nz>
Newsgroups: sci.crypt
Lines: 544

This is going to have to be one of my last. I just can't afford to spend several hours responding as I have here.

On Tue, 20 Apr 1999 20:35:53 -0400, in <371D1D69.AF9907B6@raas.co.nz>, in sci.crypt Geoff Thorpe <geoff@raas.co.nz> wrote:

>Hi,
>
>Terry Ritter wrote:
>>
>> On Tue, 20 Apr 1999 00:28:14 -0400, in <371C025E.6AD4BAB8@raas.co.nz>,
>> in sci.crypt Geoff Thorpe <geoff@raas.co.nz> wrote:
>> >

>[...]
>Before I get accused of doing precisely what I don't want to do (lock
>horns, duke it out, etc) ... let me just say that I really am warming to
>an idea implicit in all of this - and I believe it is one of yours,

Son of a gun, it *is* one of mine. Why, what a surprise!

The first thing that happens in these discussions is the outright denial of my points. Then, as my points become unassailable, there is denial that I was the one who presented solutions -- in the very same discussion! Then we will have denial that I originated this issue, then condescending comments that I was not the first to ever do so (despite the level of controversy implying that earlier discussions had little effect). And then, since I talk about this periodically, we will have comments it has to be considered public domain anyway.

Is it any wonder that I patent my stuff?

The current relevant message was:

From: ritter@io.com (Terry Ritter)
Newsgroups: sci.crypt
Subject: Re: Thought question: why do public ciphers use only simple ops like shift and XOR?
Date: Fri, 16 Apr 1999 20:20:24 GMT
Lines: 129
Message-ID: &LT:37179b67.12809750@news.io.com>

but we can go back *years* for much of this.

>though it was Trevor I think who recently illustrated it quite well ...
>namely the employment of a standard bank of ciphers that can be invoked
>on demand in any number of possible configurations eg strung together in
>a different order every time, utilising the different modes of
>operation, etc etc. I also agree the implementation and standardisation
>headaches of this sort of scheme are not insurmountable - indeed every
>standard SSL implementation I've seen lately seems to implement most of
>the core ciphers/modes that could be used in such a scheme. I'm also
>definately not against the idea of extensibility of frameworks to
>incorporate as-yet-unknown elements - indeed PKCS#7 probably didn't have
>DSA, ElGamal etc in mind, but now they seem to be creeping into CMS and
>that method seems to allow room to grow it again later. (If I've
>confused this with something else, someone please correct me - I could
>have my wires a little crossed right now and don't have any reference
>handy).
>
>But it seems to me, especially with regard to non-realtime applications,
>that to an extent, less-is-more ... sure a few ciphers in your pool is
>fine, especially if everyone has them. But the wholesale liberalisation
>of
>cipher farming seems to create a very real problem - a kind of protocol
>grid-lock. And frankly, I still place a lot of stock in what *I* rank as
>ciphers of tested strength and wouldn't want any system of mine having
>too many "new toy" ciphers creeping in. Perhaps we need to agree to
>disagree.

I'm *sure* we will disagree. But if your disagreement is with the above issues, you again disagree with your particular extrapolation of such a system. Let me describe the right way:

First, we want to be able to plug in arbitrary ciphers. The interface thus has several levels: One level is the OS interface which allows each of multiple packaged routines to be stored on disk, then dynamically loaded and invoked. Another level is the parameter package and functioning of each routine.

Next, we want to be able to accommodate essentially unlimited future ciphers, and do so in a way which does *not* require a central registration facility (which thus must be operated and funded), with its inherent submission, approval, and listing delays. We can do this by having the implementor of each package give it a unique textual name -- perhaps the company name, style, model number (NOT a serial number). This would be a string which in practice would probably be a single line of 80 characters. When we want that cipher, we use its textual name. Presumably, the cipher system will catalog the various packages available by name so they can be quickly loaded from disk then invoked.

Then we want to satisfy users desire for particular ciphers, or to *not* use particular ciphers. We can do this by generating a list of ciphers which each user will accept -- a different list for each user, and each connection that user has. We assume that everybody has one of these ciphers, probably 3-key Triple-DES. (In general, once this gets rolling, almost everybody will have the basic 10 or 20 ciphers.)

Next, we want to support changing ciphers mid-conversation. We can do this by establishing a "control channel" which is just a variable-length field distinguished from the normal data payload. All of this is enciphered for transmission, then deciphered. On the control channel, the ciphers propose a short list ciphers to change to, and then additional lists if these are rejected. When they find agreement, both ends switch. I suggest that each direction have its own current cipher, and that ciphers be switched every message or two.

One implication of this is that there must exist a local encrypted database of keys and associated current cipher selections, which must be updated dynamically as the ciphers change. The user must unlock this with the single key he or she needs for the whole system.

I could go on with a specific cipher-change message protocol, but will not.

>> Your position, dare I state it, is that you *can* estimate the
>> capabilities of your Opponents. You also say you can estimate the
>> future strength of a cipher from past tests. But for all this
>> claiming, we see no similar statements in the scientific literature.
>> So these are simply your opinions, and I see no supporting facts.
>
>Scientific literature? Ironic that it is precisely this quantity that
>you appear to place very little value in with regard to ("tested")
>cipher strength, and yet I am supposed to find some to support my view?

Aw, man, this is soooooo bogus. You have no idea what you are talking about.

Show me one scientific article which *does* specify cipher strength. Crypto scientists *know* they CANNOT state a "strength" (as we know the term). There *is* no (reputable) literature like this. Yet that is exactly what YOU are trying to do.

In this case I agree with virtually the entire body of cryptanalytic

literature in that one CANNOT know cipher strength. I also think it is fruitless to speculate on strength, or on the capabilities of our Opponents, and that we are better of spending our time protecting against failures which cryptanalysis cannot avoid.

Which means that attempts to do this -- exactly what you are doing -- are simply unscientific. When you can show that this works, then we can talk about it. (But that is only the beginning of your argument.)

>Anyway - I have already said that my view (that a cipher not falling
>over despite some considerable efforts against it does merit some
>"value") is not based on any exact science. I think history, and some
>basic common sense warrant my conclusions. Your contrary opinion does
>not appear to be any more scientifically founded - although it does
>appear to be a little more "absolute" or "axiomatic" (and IMHO "not
>terribly practically useful").

But my "contrary opinion" -- that the past history of the strength of a cipher does NOT tell us about its future strength -- again reflects the scientific literature. I am aware of no articles at all that show such a correlation. That is not *my* opinion, that is the prevailing scientific understanding. You are the one proposing a clear opinion with no scientific basis whatsoever.

>> >Now, statement (1) is wrong.
>>
>> Which was: "1) We cannot estimate the probability that an effective
>> attack exists which we did not find."
>>
>> Since you think this is wrong, you must believe we *can* make an
>> estimate. Fine. Do it. Show me.
>
>The fact that I can drive in Quebec without getting killed for 3 months
>suggets I can probably survive another few days. I don't know what my
>chances would be in London - and maybe the insurance salesman doesn't
>either. Fine, I'll go for a drive shortly and if I STILL don't get
>killed (ie. I post again in the future to this group) then that supports
>my estimate of the probability.

And, as I have said repeatedly, though apparently to little avail, we know the general risk of driving from reporting and experience.

In contrast, there is no reporting of crypto failure.

And we do not experience cipher failure, because simply using a cipher program does not tell us whether or not that cipher has been penetrated and our data exposed. We get no feedback upon which to build an understanding of the risk of cipher failure.

In crypto, we do not have the same cues which support our understanding of risk in real life.

>If you think I'm wrong, break triple-DES
>and you show me.

Nonsense. My point is precisely that cryptanalysis ("breaking") *cannot* tell us if a cipher is weak. My point is that we must assume weakness *without* having to break the cipher, if failure would be disastrous. Since that is my point, I hardly need do the opposite to make my argument.

>Otherwise - neither of us is correct in any pure sense

I think unscientific arguments *would* be called "incorrect." You assume something trivial like extrapolating the strength of a cipher from its cryptanalytic testing -- something which does not exist in the scientific literature.

In contrast, I assume that any cipher may fail -- and this is the nearly universal scientific understanding. I would call that correct.

>... but I'm still comfortable with my approach and if others are too
>that's all that matters.

I doubt the facts are changed by their popularity.

>Anyway, now I think about it further - exactly
>how can you possibly insist that "we cannot estimate a probability" ???
>Sounds absurd. Particularly with something that has any historical
>record at all?

ONE MORE TIME... Fine. Do it. Show me. Show us all. Show every cryptographic scientist what they have been missing. Go. Do it now.

>As someone with a love of pure mathematics, it does feel a little
>disturbing to be arguing a point with someone where it is *I* who am on
>the fuzzy, pragmatic, approximation side of the fence and the other is
>arguing puristically.

You seem to be in conflict between your ego and reality. You have

told us how good you see yourself as being, which leaves very little room to realize that your entire argument has been wrong from the beginning.

>> Alas, what people believe is not science.
>
>But what people believe influences what they will and will not do (and
>will or will not put up with). And unless a scientist can *prove*
>absolutes they will have difficulties imposing absolutes. Perhaps a good
>way to measure this is to ask an insurance-brokerage expert to comment
>on the insurability (premiums etc) on an information resource secured
>using your approach versus something like I prefer. Not a single ounce
>of "science" will enter into this equation (I suppose) and yet I can't
>imagine a more adequate way to judge the situation - after all, it is
>these kind of people whose lives it is to cover the costs of things when
>they go wrong.

Will insurance really cover the expenses of an entire society changing from one standard cipher which fails to another "better" cipher? And what do we do about the little detail that this failure may occur in secret and extend over a period of decades? Could any *country* pay such a cost? Of what worth is insurance if the reality of failure is Apocalyptic?

>> >year than the average "expected life". It's a very basic and common
>> >mathematical model/argument, and it's common sense.
>>
>> Oddly, no such study has appeared in the literature. That seems
>> somewhat strange, since you say it is very basic common sense.
>> Perhaps everyone else in cryptography has simply been blinded to this
>> fundamental truth. When will you write it up for us?
>
>If I hire a programmer to work with a new technology and a deadline, and
>my options (for the same money/conditions etc) are between someone who
>has demonstrated he/she can handle new technologies (in the past of
>course), and someone who *might* be able to handle new technologies, I'm
>going to hire the one with experience. A new candidate might be faster,
>hungrier, and actually better with the new technology - but why would I
>take that chance versus the chance the experienced one ran out of puff?
>True, until I try one I will not know which one was better but I'll hope
>you agree estimations, probabilities, and common sense are all present
>and can be utilised. I got a feel that your view on this was almost
>quantum mechanical - then I remembered that even QM admits probability
>distributions and expected values (and the difference between a likely
>result and an unlikely one even though each is possible until you find
>out for sure).
>
>But I digress perhaps, and we've already demonstrated we don't agree
>here so ...

It is more than a matter of disagreement; it is a matter of you being wrong.

>> You are arguing your opinion about cipher strength. (Recall that I do
>> not argue an *opinion* about cipher strength, but instead the *fact*
>> that any cipher may be weak.) If you have compelling factual
>> evidence, I will support it. Show me the correlation you say exists.
>> Prove it. Then you can use it.
>
>I've already admitted that my "correlation" is a fuzzy one, full of
>ideas that are "compelling" (to me) here, "suggestive" (to me) there,
>etc - and that my conclusion is a fuzzy one. Perhaps then I've shown
>compelling "fuzzy" evidence. [;-) Anyway, you are saying I cannot use
>"tested strength" as a measure - and your sole reason seems to be -
>"because it could still break tomorrow". Nobody disputes the latter
>statement but it does not logically imply the blanket assertion you
>make. Not proving things one way or the other does not mean we need
>default to your assertion, that all ciphers are equal

You oh-so-casually use my words out of context. As you use it, that was not my assertion. I claim that any two ciphers which have not failed are equal with respect to the possibility that they may fail. In that sense, all ciphers are equal, so we can trust none. All ciphers are equal in that we cannot trust them. That includes new ciphers, old ciphers, and everything in between.

>when only existing
>failures to break them are in evidence, and abandon my assertion, that
>failing to break ciphers does provide useful information for
>"estimations".
>
>And in case you ask, no - I know of NO research paper to support this
>and have no interest in attempting to create some when I'm already
>satisfied.

I would simply hope that your readers are more scientific.

>> Nobody has any problem with you making a call for yourself and risking
>> only yourself. But if this "call" is intended to formulate what
>> "should" happen for much of society, you may need to revise your
>> estimate as to the consequences of failure. Just how much disaster

>> are you willing for us to have?
>
>The *consequences* of failure are not what I'm estimating. And again,
>I'll agree that the idea discussed before (utilising a defined set - for
>interoperability this seems necessary - of ciphers, algorithms, etc etc
>that can be jumbled around on the fly to diffuse the impact "a break"
>would have). It would interesting, though off topic, to see how your
>absolutist approach generalises to arms control, transportation
>legislation, etc. All areas where "pragmatic fuzzies" tend to preside
>over "puristic absolutes" - even when they're cautionary variety.

I have no idea what you are talking about. Those are not analogies I use; I doubt they apply to crypto. If you want to draw such analogies, you will have to do so in much greater detail.

>> Will it be OK for everyone to use the single standard cipher which you
>> predict is strong, if you turn out to be wrong? Will it be OK when
>
>I've already moved a bit to your side on at least one point - one single
>cipher (if they are implicitly atomic and cannot encompass the idea that
>one can effectively put what would be 3 or 4 atomic ciphers into a
>"cipher") would not be as comforting as a small (I still think "fixed",
>or at least "slow moving") collection of ciphers jumbled up to disperse
>the impact a break in any one configuration would have. I still think my
>point applies to the selection of those ciphers though.

Who knows, you may move "a bit" on that, next.

>> communications grind to a halt and incompatible low-security temporary
>> measures are instituted everywhere while a new cipher is integrated
>> into all the programs which must be replaced throughout society? Is
>> that OK with you?
>
>And quantum computers could break everything and that wouldn't be OK
>with me either. But I'm not going to resort to carrier pigeons (which
>could be broken by a large society of hunters ... oh god ... this is
>getting too much).

I would say so.

>> >Our Opponents are just well-paid versions of us, most of
>> >whom probably grew up around us, and who find their occupations not too
>> >unfathomably unethical to suffer day by day.
>>
>> This is simply breathtaking: It is appallingly, plainly, immediately
>> false even to the most casual observer. People do differ, we do have
>> limitations others do not have, and others often do take advantage of
>> knowledge to which we have no access.
>
>You still don't get what I'm saying ... YES people do differ, but I
>think continuously, not by quantum leaps that erase any relationship you
>can draw.

And I think that is false.

>> >Sure thing - but the whole system does not collapse down to a binary
>> >system of "broken" and "not-broken-yet" ... as you say, you put together
>> >a threat model ... consistent with your requirements and using a chosen
>> >method for judging a components "worth", and amplify it here and there
>> >as appropriate. A lot like putting together a cost-proposal I guess ...
>> >add in your known prices, choose an acceptable value for the "unknowns",
>> >amplify the costs of all the "risky" bits, add x% profit on top - and
>> >then bang another 30% on top for good measure, and generally covering
>> >your butt some more.
>>
>> Write whatever numbers you want: you cannot support them.
>
>You can be as cautious as you like and you could still get busted - you
>can be as irresponsible as you like and you COULD (not likely) get away
>with it. You can also just give up. That same model applies every time I
>write a proposal, an electricity company designs and insures an
>infrastructure, and many other real world situations. Tell me why I HAVE
>to resort to such a binary system of "broken" and "not-broken-yet".

I wouldn't know.

>You
>don't seem to be able to support your claim that the test of time (and
>attack) does not provide a usable measure and you yourself have not
>written any numbers to try.

My support is that the "test of time" theory does not exist in cryptographic science. I suggest that the lack of results from a great many very smart people is precisely the sort of prediction you claim to support. And now you ignore it because it is not to your advantage.

>on't tell me and many other people with an
>interest that it's invalid to use such approaches, and then only support
>your claim by statement - particularly if you intend to then insist I
>support my own claims with numbers or proofs I'm supposed to pluck out

>of thin-air.

My claims are supported by the literature, and yours are not. You thus have the greater burden. How unfair of me to be on the right side.

>> >3 ciphers strung in a line is, to me, a cipher.
>>
>> The distinction is that each cipher is an independent and separable
>> element which can be "mixed and matched" with any other. Each cipher
>> is tested as an independent unit, and brings whatever strength it has
>> independent of internal ciphering requirements. Dynamic mixing and
>> matching prevents having any fixed target to attack.
>
>So should good cipher design as far as I can see but I'll go along with
>you here. I see this idea as promising and will not argue with the
>premise that if you've got 5 good ones, why just stick with one - indeed
>why just stick with a fixed arrangement of that 5 (effectively making
>one very complicated, but still fixed, cipher) when you can jumble the
>order, modes of operation, etc each time. (The way in which that has
>been done would presumably become part of the "key"). I'd still prefer
>that we standardise on those 5 and maybe rotate new ones in
>"occasionally" (conservatively) in a method not-unlike the current AES
>process - ie. public exposure to candidates for a good hard thrash at
>them before actual incorporation of them into systems.
>
>> >You need all three in
>> >the same place and in the same order to have anything other than a
>> >"noise generator". Breaking 3 ciphers should be no more difficult than
>> >breaking one well designed one using 3 different stages
>>
>> Really? Tell me more about how ciphers are designed. How long did
>> you say you have been doing this? How many ciphers have you designed?
>> Have you measured them? Where can we see your work?
>
>Already told you I'm not a cipher designer. But there are cipher
>designers who share my view so attack the idea, not the guy saying it.

Which particular idea would that be, precisely? Will you find a cipher designer who will say in so many words that he or she *can* predict the strength of a cipher based on its cryptanalytic history? Will you find someone who will say they can predict the capabilities of future Opponents based on current academic results? I don't think so.

What you actually can find is people who will say that untested ciphers are likely to fail. Note that this is distinctly different from your claims. In particular, having "passed," tells us *nothing* about the strength with respect to unknown attacks.

I support all the cryptanalysis we can get. But, in the end, we can trust neither cipher to remain unbroken. So, in that sense, new and old ciphers are both equal in their untrustability.

>I
>might also add - you're asking me to measure ciphers after having
>insisted quite strongly that any such attempt is implicitly impossible
>(with the absolute exception of breaking it).

You are the one who keeps claiming that realistic comparisons of cipher strength are possible. That is not in the literature, so it is up to you to back it up your claim. Show me. Do it.

I claim it is not possible, and I have the literature on my side.

>Can I take apart a modern cipher and say "that's not a cipher - look,
>it's lots of little ciphers"? All I said was the division for me between
>3 ciphers strung in a line and one cipher with 3 stages to it seems to
>be a question of packaging and patents. One could even stretch the
>definition and include the possibility of reording "stages" based on the
>"key". But I'm not going to suck myself into a bits-on-the-wire cipher
>designing discussion because I know I can't make a worthwhile
>contribution to it.
>
>> >regular basis. I see this as unacceptable in a real-world scenario for
>> >reasons of interoperability & standardisation, as well as security.
>>
>> What you mean is that *you* do not know how such a system would be
>> standardized and made interoperable. Then you imply this means
>> *nobody* *else* could do it either.
>
>Fair call. Let me try again then - I think there could well be some very
>useful gains made from employing standards that use multiple primitives
>that hopefully seem (barring quantum computers?) to be independant
>targets of attack, that when used in different orders, modes, etc reduce
>the chances of the whole system getting broken rather than one putting
>the house on one primitive, or one configuration of the primatives. I do
>however think we should be measuring the primitives (which you suggest
>is pointless) as best we can, and that we should use a conservative
>approach to the standardisation on those primitives and the method by
>which new ones are incorporated into the standards.
>
>If your "boolean model" of cipher strength is valid - can't this entire

>idea, when wrapped and considered as an entity in itself, then be
>implicated as just as "trust-worthy" as a single cipher that hasn't been
>broken? I would NOT regard them as equal but your argument, by
>extension, does.

Without going overboard, I agree! I think "my" Boolean model is overstated, though: Ciphers can be "Boolean" in an abstract sense of broken or not. But since we cannot measure either cipher strength or Opponent capabilities, the distinction seems a waste of time.

We can no more have absolute "trust" or absolute "confidence" in the strength of a layered system than any one cipher. But what I think we *can* say is: 1) the stack is not weaker than any of the components, and 2) the stack prevents single-component failure from being an overall failure. We might *speculate* that this "lessens" the probability of failure. But since we cannot measure any of these strengths or probabilities, that seems like yet another chimera just better ignored.

Terry Ritter ritter@io.com http://www.io.com/~ritter/
Crypto Glossary http://www.io.com/~ritter/GLOSSARY.HTM

Subject: Re: Question on confidence derived from cryptanalysis.
Date: Wed, 21 Apr 1999 11:51:36 -0400
From: Geoff Thorpe <geoff@raas.co.nz>
Message-ID: <371DF408.C995EE10@raas.co.nz>
References: <371d7e0a.11538318@news.io.com>
Newsgroups: sci.crypt
Lines: 185

Hi,

I'm clipping liberally now in part of a joint effort (with Terry I believe) to get this down to a managable size again. Particularly as the themes seem quite universal throughout.

First let me quote the more "emotive" stuff before getting the heart of the issue itself:

Terry Ritter wrote:
> You seem to be in conflict between your ego and reality. You have

OK, so now I have an ego problem - after caveats of not being a cipher-designer, knowing that my views of "tested strength" are based on "fuzzy" quantities and "judgement". But then you go on to demonstrate some hypocrisy:

> told us how good you see yourself as being, which leaves very little
> room to realize that your entire argument has been wrong from the
> beginning.

how arrogant, and yet somehow sad too.

> Aw, man, this is soooooo bogus. You have no idea what you are talking
> about.

once again.

> Son of a gun, it *is* one of mine. Why, what a surprise!
>
> The first thing that happens in these discussions is the outright
> denial of my points. Then, as my points become unassailable, there is
> denial that I was the one who presented solutions -- in the very same
> discussion! Then we will have denial that I originated this issue,
> then condescending comments that I was not the first to ever do so
> (despite the level of controversy implying that earlier discussions
> had little effect). And then, since I talk about this periodically,
> we will have comments it has to be considered public domain anyway.
>
> Is it any wonder that I patent my stuff?

Is it any wonder you so acidically slam views that do not concur with your own - I'm not rallying behind anything other than my opinion - and I am impartial of the quantities we're dealing with here, I maintain no patents or profit - I do not get paid to do anything "free" (as per the academics) - and yet what I do contribute (coding), in the area of crypto at least, goes straight into the public domain. I stand accused of getting stuck between ego and reality, but all I've done is state that I don't agree with your radical view on the value (or lack of value) one can put on "tested strength" - you however seem guilty of exactly that which you accuse me of. At least I've engaged in discussion with you, it seems your beef should be with those who pay no attention to you.

> I'm *sure* we will disagree. But if your disagreement is with the
> above issues, you again disagree with your particular extrapolation of
> such a system. Let me describe the right way:

"the right way" ... why am I not surprised.

[snipped a description of a basic and extensible architecture for a user-maintained collection of ciphers and an outline of how the online

protocols might proceed from that].

Frankly Terry there are much better people to comment on this from audit and cryptanalytic points of view but from a software engineer/designer point of view, really nothing here surprises me a great deal. My initial reaction is that it looks a little bit held together with "chicken-wire", and you already know my point of view on standards that just say "plug in the ciphers you feel most 'connected' to" - I'm still not compelled that this will be interoperable but that's not to say you don't have an explanation why it could be - and I'm certainly compelled that this is irresponsible in the extreme, but that of course hinges on my view that triple-DES is a better option than mysticTarot128 even though the latter could have a sexier web-site. And we already know we disagree on that premise so it boils down to axiomatic differences.

> I could go on with a specific cipher-change message protocol, but will
> not.

Might I just say that it seems to me that this approach (a) seems to demand a complicated protocol that itself must be a vulnerable-looking target for a "winner-takes-all" breakage, (b) if you want to to sling requirements of "scientific literature" around then why don't we quote the oft-quoted phrase "security by obscurity". I would still rather use one 128-bit triple-DES stage, than two 128-bit toys in a random configuration. The latter looks more like snake-oil to me and may well impress end-users (for being "configurable" and "too complicated to break (TM)") and impress Opponents ("ha, the fools"). Again, this could boil down to our fundamental difference.

> Show me one scientific article which *does* specify cipher strength.
> Crypto scientists *know* they CANNOT state a "strength" (as we know
> the term). There *is* no (reputable) literature like this. Yet that
> is exactly what YOU are trying to do.

You go on to say that you welcome all the concerted cryptanalysis people can come up with. For what? Is that cryptanalysis worthless if it does not actually break a cipher? If not, tell me what value you place on such cryptanalysis (let's say the analysis in question is on DES/triple-DES), call it a unit of "tested strength", and perhaps we don't disagree as much as we did.

> In this case I agree with virtually the entire body of cryptanalytic
> literature in that one CANNOT know cipher strength. I also think it

Yet much of it fails to break the cipher in question, and is often littered with conclusions such as "seems to hold up well in the face of [*]", "seems to have some strong properties with respect to [*]", etc. These are morsels that contribute to what I perceive as "tested strength". You seem to think that cryptanalysis is valuable, and yet you place no value on most of it. I choose to. And you also say that the entire body of cryptanalytic literature supports you in all this and it is I who must find evidence, proof, references to support my disagreements with you - when what I'm saying is that cryptanalytic work against a cipher (that doesn't bust it) gives me some confidence over a lack of cryptanalytic work against another. You've still not convinced me that I must abandon that view - you've just stated that I should, and that the literature supports you in that conclusion.

> is fruitless to speculate on strength, or on the capabilities of our
> Opponents, and that we are better of spending our time protecting
> against failures which cryptanalysis cannot avoid.

That is a noble objective - but whatever the result, it will employ ciphers - and that's where my niggly (and probably highly frustrating to some obsessed with an all-ciphers-are-equal philosophy) little view comes back into the frame.

> Which means that attempts to do this -- exactly what you are doing --
> are simply unscientific. When you can show that this works, then we

Define "scientific" and we'll probably see you've defined the possibility of discussing this issue scientifically out of existence.

> But my "contrary opinion" -- that the past history of the strength of
> a cipher does NOT tell us about its future strength -- again reflects
> the scientific literature. I am aware of no articles at all that show

Does it indeed. Funny that rather prolific contributors to the scientific literature are competing in a battle to see whose cipher holds up the best to "historical strength testing" so as to be utilised with *improved* expectations of "future strength". Perhaps these luminaries won't come out and argue the point with you, but because I'm not so highly esteemed and you're here arguing with me - naivety, unfamiliarity with the literature, historical record, blah blah blah are all valid accusations for you to dismiss the view outright.

> such a correlation. That is not *my* opinion, that is the prevailing
> scientific understanding. You are the one proposing a clear opinion
> with no scientific basis whatsoever.

The opinion that placing "tested strength" in something that has withstanded attempts to break it over things which haven't is a core scientific principle in many fields, many of them where disastrous risks of being wrong are involved - and it IS common sense. Telling me it isn't does seem to put a burden of proof on you that goes beyond simply

stating it, and making sweeping assertions that "the vast scientific literature" supports you.

> Nonsense. My point is precisely that cryptanalysis ("breaking")
> *cannot* tell us if a cipher is weak. My point is that we must assume

And apparently cryptanalysis ("not breaking") *cannot* tell us if a cipher is "strong". What value is it that you actually see in this field of science? How are anyone except the scientists themselves supposed to use or apply the outcome of that work in any practical way?

> I think unscientific arguments *would* be called "incorrect." You
> assume something trivial like extrapolating the strength of a cipher
> from its cryptanalytic testing -- something which does not exist in
> the scientific literature.

Common-sense usage of scientific literature itself not being documented in scientific literature. An interesting rebuttal and one that has me tiring of this pointless back-and-forth. If you are blind, this will go nowhere - if you are right, you need to find a better way of understanding my view and showing me constructively why it is definitively wrong if you want to get anywhere. Bear in mind that my view happens to be shared by many who can not so trivially be swept aside with back-handed commentary about "the scientific literature" and "not knowing what you're talking about".

Regards,
Geoff

Subject: Re: Question on confidence derived from cryptanalysis.
Date: Thu, 22 Apr 1999 00:21:12 GMT
From: jsavard@tenMAPSONeerf.edmonton.ab.ca (John Savard)
Message-ID: <371e664f.28640885@news.prosurfr.com>
References: <371d7e0a.11538318@news.io.com>
Newsgroups: sci.crypt
Lines: 85

ritter@io.com (Terry Ritter) wrote, in part:
>On Tue, 20 Apr 1999 20:35:53 -0400, in <371D1D69.AF9907B6@raas.co.nz>,
>in sci.crypt Geoff Thorpe <geoff@raas.co.nz> wrote:

>>Before I get accused of doing precisely what I don't want to do (lock
>>horns, duke it out, etc) ... let me just say that I really am warming to
>>an idea implicit in all of this - and I believe it is one of yours,

>Son of a gun, it *is* one of mine. Why, what a surprise!

>>though it was Trevor I think who recently illustrated it quite well ...
>>namely the employment of a standard bank of ciphers that can be invoked
>>on demand in any number of possible configurations eg strung together in
>>a different order every time, utilising the different modes of
>>operation, etc etc.

I've caught myself stealing some of your ideas - usually in E-mail discussions - and although I suspect what I've done in Mishmash (see Quadibloc III) isn't really the same idea, it uses _part_ of it, in a limited form so as to fit within the framework of a "conventional" block cipher.

>>And frankly, I still place a lot of stock in what *I* rank as
>>ciphers of tested strength and wouldn't want any system of mine having
>>too many "new toy" ciphers creeping in. Perhaps we need to agree to
>>disagree.

>First, we want to be able to plug in arbitrary ciphers.

>Next, we want to be able to accommodate essentially unlimited future
>ciphers, and do so in a way which does *not* require a central
>registration facility (which thus must be operated and funded), with
>its inherent submission, approval, and listing delays.

>Then we want to satisfy users desire for particular ciphers, or to
>*not* use particular ciphers.

Which is indeed the point where agreement is restored.

>Next, we want to support changing ciphers mid-conversation.

This makes it clear enough that you are in a different and more advanced realm than what I was worried about having "stolen".

While there will be pressure to adopt the standard ciphers for interoperability, such a system definitely does allow for a new cipher to become a new part of the set.

>We can no more have absolute "trust" or absolute "confidence" in the
>strength of a layered system than any one cipher. But what I think we
>*can* say is: 1) the stack is not weaker than any of the components,
>and 2) the stack prevents single-component failure from being an
>overall failure. We might *speculate* that this "lessens" the
>probability of failure. But since we cannot measure any of these
>strengths or probabilities, that seems like yet another chimera just
>better ignored.

Unless we fall for that chimera - unless we believe that adopting a layered system will do some good, and reducing the probability that our messages will be read is indeed the only goal pursued here - why bother? Of course, (1) and (2) are valuable in themselves: essentially, (2) is worthwhile pursuing because of the _possibility_ (absent any provably existing nonzero probability) that it will _reduce the probability of failure_.

Even when no progress towards a goal can be proven to have taken place, it is impossible to avoid, however chimerical it may be, evaluating measures taken to achieve a goal in terms of that goal. Even when all we have are possibilities instead of probabilities.

And just as using a layered system doesn't *prove* anything, so does using ciphers that have been studied and found to be resistant against a variety of known attacks. Yet it seems like a sensible thing to do, for want of anything better.

Using a secret cipher of your own for your own communications *also* makes sense, for different reasons, and using the latest and greatest design, not very well tested yet, because it has a larger key size also makes some sense, and so does using an obscure cipher that attackers may not have bothered with. Because there _are_ different "sensible" things to do than are necessarily popular or respectable - and more sensible things to do than any one cipher can embody - the layered use of multiple ciphers is a good idea. Even if it proves nothing.

John Savard (teneerf<-)
http://members.xoom.com/quadibloc/index.html

Subject: Re: Question on confidence derived from cryptanalysis.
Date: Tue, 20 Apr 1999 18:50:48 -0700
From: Jim Gillogly <jim@acm.org>
Message-ID: <371D2EF8.34DC6802@acm.org>
References: <371c15e3.8290372@news.io.com>
<371C025E.6AD4BAB8@raas.co.nz>
Newsgroups: sci.crypt
Lines: 71

I think Terry Ritter's right to be concerned about having essentially everyone move to a single new cipher. If the danger isn't obvious, consider the analogy with biological systems, where a species with no genetic diversity can be wiped out by a single virus incident. Or with computer systems, where something like Melissa can cause widespread annoyance and some down time because almost everyone is using the same operating system and office software suite.

I also agree with him that a careful concatenation of ciphers can help limit the damage. I think we may disagree on what kinds of ciphers would be most appropriate as choices for concatenation, since I prefer ciphers that good analysts have tried and failed to break over ciphers that nobody with cryptanalytical experience has looked at. I define a good analyst as someone who has broken a difficult system.

However, I (like John Savard) think Terry overstates some issues. Here's a case in point:

Terry Ritter wrote:
> Your position, dare I state it, is that you *can* estimate the
> capabilities of your Opponents.

In another article he wrote:
> But the only thing being "measured" here is the open, academic
> analysis. The *real* experts do not play this way. We thus have no
> way to understand their capabilities. The strength value measured on
> academics cannot apply to the real problem.

These and similar remarks suggest that a conservative threat analysis must regard the opponents as god-like in their cryptanalytic capabilities. Of course in the limit this isn't useful, since we would have no more confidence in a concatenation of ciphers against an opponent like this than we would in a single cipher.

However, we do have ways to estimate the capabilities of the opponents. I suggest that the government cryptologic agencies of the US and UK represent conservative surrogates for the cryptological skills of the strongest opponents, and we have seen several unclassified examples of times when they were less than perfect.

In one case (factoring circa 1973) the UK agency was no further advanced than the academic community, and academic advances in that field were made shortly thereafter. In two other cases the US agency made embarrassingly public blunders (the Clipper checksum exploited by Matt Blaze, and the SHA/SHA-1 botch that they noticed and fixed themselves) that would not have been made if they were omniscient. I don't include Biham's work suggesting SKIPJACK is not a conservative design, since we don't know that it has to be -- for all we know, there are wads of supporting theorems that it's precisely as strong as it needs to be for its size. We do have a couple of other cases of classified discoveries and corresponding unclassified ones: IBM's differential cryptanalysis (15 years) and CESG's non-secret encryption (4 years). There are also training exercises (the Zendian Problem and a British

special intelligence course) which anyone can use to compare their skills with advanced cipher school students of the 1960s. The latter does not, of course, give the peak strength of the best cryppies, but does suggest a starting point for the curve. Finally, we have retired NSA cryppie Robert H. Morris's remarks at Crypto '95, where he said that by the middle to late 1960's cryptanalysis had become less cost-effective than other methods of gaining the information. One may choose to disbelieve him, but I don't.

In any case, we do have some data points on the capabilities of the strongest potential opponents, and assuming they're perfect would be overly conservative.

--
Jim Gillogly
30 Astron S.R. 1999, 00:51
12.19.6.2.5, 1 Chicchan 13 Pop, Ninth Lord of Night

Subject: Re: Question on confidence derived from cryptanalysis.
Date: Wed, 21 Apr 1999 05:04:20 GMT
From: ritter@io.com (Terry Ritter)
Message-ID: <371d5c4c.2898786@news.io.com>
References: <371D2EF8.34DC6802@acm.org>
Newsgroups: sci.crypt
Lines: 153

On Tue, 20 Apr 1999 18:50:48 -0700, in <371D2EF8.34DC6802@acm.org>, in sci.crypt Jim Gillogly <jim@acm.org> wrote:

>I think Terry Ritter's right to be concerned about having essentially
>everyone move to a single new cipher. If the danger isn't obvious,
>consider the analogy with biological systems, where a species with no
>genetic diversity can be wiped out by a single virus incident. Or with
>computer systems, where something like Melissa can cause widespread
>annoyance and some down time because almost everyone is using the
>same operating system and office software suite.

I think I have a right to cheer at this agreement with my major point.

>I also agree with him that a careful concatenation of ciphers can
>help limit the damage.

And then I cheer again at this agreement with part of my proposed solution package.

>I think we may disagree on what kinds of
>ciphers would be most appropriate as choices for concatenation,
>since I prefer ciphers that good analysts have tried and failed to
>break over ciphers that nobody with cryptanalytical experience has
>looked at. I define a good analyst as someone who has broken a
>difficult system.

Then I assume you are willing to make the services of such an analyst available free of charge and without delay. The way it is now, one cannot get such analysis unless one is a particular type of person, working in a few selected environments, and with particular types of design. Having inherited a democracy, I am unwilling to give that up for supposed advantages which, in the limit, do not give us what we want anyway. I think people should be able to select their own ciphers based on any criteria they want, including superstition and innuendo.

>However, I (like John Savard) think Terry overstates some issues.
>Here's a case in point:
>
>Terry Ritter wrote:
>> Your position, dare I state it, is that you *can* estimate the
>> capabilities of your Opponents.
>
>In another article he wrote:
>> But the only thing being "measured" here is the open, academic
>> analysis. The *real* experts do not play this way. We thus have no
>> way to understand their capabilities. The strength value measured on
>> academics cannot apply to the real problem.
>
>These and similar remarks suggest that a conservative threat analysis
>must regard the opponents as god-like in their cryptanalytic
>capabilities.

If that is what you take from these comments (in their proper context), I am not surprised that you call my position overstated. However, you have exaggerated my position.

In particular, I doubt I have ever said the Opponents are "god-like." As far as I can recall, the only people I have accused of being "god-like" are the crypto gods who seem to be able to predict: 1) the future strength of a cipher, based on past tests; and 2) the capabilities of unknown Opponents, based on the capabilities of known

academics.

>Of course in the limit this isn't useful, since we
>would have no more confidence in a concatenation of ciphers against
>an opponent like this than we would in a single cipher.

And so, clearly, I do not so assume. Since I do not assume that an Opponent has unlimited capabilities, this comment strongly misrepresents my arguments.

But what *are* we to assume? Even a *modest* "value" for Opponent capabilities is also "not useful" to us. This is because it is (virtually) impossible to *measure* knowledge, experience, and innovation. And then it is impossible to *measure* cipher strength. So we first don't know the difficulty of the problem, and then don't know the capabilities our Opponents can bring to the solution. This naturally leave us in a quandary, even *without* assuming unlimited capabilities. The problem is *not* that we should assume reasonable value for Opponent capabilities, the problem is that *any* such values and their implications are unknown, uncalibrated, and unuseful.

I suggest that this whole line of inquiry (into cipher strength and Opponent strength) is a waste of time. Since we know that single-cipher failures are possible, we can work to fix that. Since I assume the triple-cipher scheme will work, it is clear that I do not assume unlimited Opponent capabilities. I do assume that whatever capabilities they do have will be stressed far harder with multi-ciphering than single ciphering. I think this is a reasonable assumption.

Moreover, by using a wide variety of ciphers, we act to limit the amount of data disclosed by any break that does occur. I do assume that this will reduce the attraction of cryptanalysis, by limiting the eventual payoff. Again, I think this a reasonable assumption.

>However, we do have ways to estimate the capabilities of the opponents.
>I suggest that the government cryptologic agencies of the US and UK
>represent conservative surrogates for the cryptological skills of the
>strongest opponents, and we have seen several unclassified examples of
>times when they were less than perfect.
>
>In one case (factoring circa 1973) the UK agency was no further
>advanced than the academic community, and academic advances in that
>field were made shortly thereafter. In two other cases the US agency
>made embarrassingly public blunders (the Clipper checksum exploited
>by Matt Blaze, and the SHA/SHA-1 botch that they noticed and fixed
>themselves) that would not have been made if they were omniscient.
>I don't include Biham's work suggesting SKIPJACK is not a conservative
>design, since we don't know that it has to be -- for all we know, there
>are wads of supporting theorems that it's precisely as strong as it needs
>to be for its size. We do have a couple of other cases of classified
>discoveries and corresponding unclassified ones: IBM's differential
>cryptanalysis (15 years) and CESG's non-secret encryption (4 years).
>There are also training exercises (the Zendian Problem and a British
>special intelligence course) which anyone can use to compare their skills
>with advanced cipher school students of the 1960s. The latter does not,
>of course, give the peak strength of the best cryppies, but does suggest
>a starting point for the curve. Finally, we have retired NSA cryppie
>Robert H. Morris's remarks at Crypto '95, where he said that by the
>middle to late 1960's cryptanalysis had become less cost-effective than
>other methods of gaining the information. One may choose to disbelieve
>him, but I don't.
>
>In any case, we do have some data points on the capabilities of the
>strongest potential opponents, and assuming they're perfect would be
>overly conservative.

First, none of this tells us about the future. Yet all operation of a cipher takes place in the future, after that cipher is designed. Unless we have a reasonable way to predict future capabilities, we are necessarily forced into conservative measures.

Next, I think it is dangerous to assume our Opponents are the intelligence services we know. In another message I suggested that if the problem was only NSA (the way it is now), we would not have much of a problem. But NSA is only an *example* of an Opponent, and not necessarily even the most advanced example in particular areas of the technology. We having intractable problems in making any serious extrapolations from this data. Again I suggest that this avenue is both unfruitful and dangerous.

Terry Ritter ritter@io.com http://www.io.com/~ritter/
Crypto Glossary http://www.io.com/~ritter/GLOSSARY.HTM

Subject: Re: Question on confidence derived from cryptanalysis.
Date: Tue, 20 Apr 1999 22:31:22 -0700
From: Jim Gillogly <jim@acm.org>
Message-ID: <371D62AA.3EDA2C55@acm.org>
References: <371d5c4c.2898786@news.io.com>
Newsgroups: sci.crypt
Lines: 30

Terry Ritter wrote:
> Jim Gillogly <jim@acm.org> wrote:
> > I prefer ciphers that good analysts have tried and failed to
> >break over ciphers that nobody with cryptanalytical experience has
> >looked at. I define a good analyst as someone who has broken a
> >difficult system.
>
> Then I assume you are willing to make the services of such an analyst
> available free of charge and without delay. The way it is now, one
> cannot get such analysis unless one is a particular type of person,
> working in a few selected environments, and with particular types of
> design.

No, I'm not. Just as you have the right to patent and profit from
your ideas, an analyst has the right to choose what she's going to
work on and how much she charges for it. If she'd prefer to spend
her time analyzing Rijndael than RC6 because the former is going to
be freely usable in her projects whether or not it's selected as the
AES, more power to her. We all make choices depending on the
outcomes we want or expect. In order to encourage more analysis one
could hire appropriate experts (as several crypto developers have
done) or offer rewards for interesting analysis whether or not it
breaks the algorithm (as I think the Twofish people have done).
But you can't expect to get expert analysis for free... the people
who chose to enter the AES bake-off aren't getting it free either.

--
        Jim Gillogly
        30 Astron S.R. 1999, 05:22
        12.19.6.2.5, 1 Chicchan 13 Pop, Ninth Lord of Night

Subject: Re: Question on confidence derived from cryptanalysis.
Date: Wed, 21 Apr 1999 07:40:34 GMT
From: ritter@io.com (Terry Ritter)
Message-ID: <371d8016.12062634@news.io.com>
References: <371D62AA.3EDA2C55@acm.org>
Newsgroups: sci.crypt
Lines: 57

On Tue, 20 Apr 1999 22:31:22 -0700, in <371D62AA.3EDA2C55@acm.org>, in
sci.crypt Jim Gillogly <jim@acm.org> wrote:

>Terry Ritter wrote:
>> Jim Gillogly <jim@acm.org> wrote:
>> > I prefer ciphers that good analysts have tried and failed to
>> >break over ciphers that nobody with cryptanalytical experience has
>> >looked at. I define a good analyst as someone who has broken a
>> >difficult system.
>>
>> Then I assume you are willing to make the services of such an analyst
>> available free of charge and without delay. The way it is now, one
>> cannot get such analysis unless one is a particular type of person,
>> working in a few selected environments, and with particular types of
>> design.
>
>No, I'm not. Just as you have the right to patent and profit from
>your ideas, an analyst has the right to choose what she's going to
>work on and how much she charges for it. If she'd prefer to spend
>her time analyzing Rijndael than RC6 because the former is going to
>be freely usable in her projects whether or not it's selected as the
>AES, more power to her. We all make choices depending on the
>outcomes we want or expect.

In that case you should agree that each user should have a similar
power to make their own choices of cipher. That sounds just fine to
me.

Of course the benefits of compartmentalizing data under different
ciphers do not really hit home until we have quite a few ciphers. And
the benefits of requiring the Opponents to "keep up" also imply a
growing substantial body of ciphers.

>In order to encourage more analysis one
>could hire appropriate experts (as several crypto developers have
>done)

Then we have the situation of reporting "scientific" results paid for
by the company which hopes to profit from those results. Will we
really trust that process? I wouldn't.

>or offer rewards for interesting analysis whether or not it
>breaks the algorithm (as I think the Twofish people have done).
>But you can't expect to get expert analysis for free... the people
>who chose to enter the AES bake-off aren't getting it free either.

I think they will not get nearly as much as they should. Which does
not mean that we do not offer them to the users just because they have
not met our desired analysis levels.

Terry Ritter ritter@io.com http://www.io.com/~ritter/
Crypto Glossary http://www.io.com/~ritter/GLOSSARY.HTM

Subject: Re: Question on confidence derived from cryptanalysis.
Date: Wed, 21 Apr 1999 08:16:39 -0700
From: Jim Gillogly <jim@acm.org>
Message-ID: <371DEBD7.A9916520@acm.org>
References: <371d8016.12062634@news.io.com>
Newsgroups: sci.crypt
Lines: 21

Terry Ritter wrote:
> In that case you should agree that each user should have a similar
> power to make their own choices of cipher. That sounds just fine to
> me.

Let a thousand flowers bloom, eh? With only 30 competent bees, many of
your flowers aren't going to get adequately pollinated.

If my banker "makes his own choice" of OTP because he read in AC that it's
unbreakable and he chooses an implementation that's easy to use since it
needs no key management, I'm the one who takes it in the shorts because
he didn't understand anything about cryptology. I as a customer don't in
general know what's being used to cover my assets, and he as a user doesn't
in general know what makes a cipher suitable for his threat model.

We have wider areas of agreement than disagreement; I'm happy to leave
it at that.
--
Jim Gillogly
30 Astron S.R. 1999, 15:00
12.19.6.2.5, 1 Chicchan 13 Pop, Ninth Lord of Night

Subject: Re: Question on confidence derived from cryptanalysis.
Date: Wed, 21 Apr 1999 19:44:12 -0400
From: "Trevor Jackson, III" <fullmoon@aspi.net>
Message-ID: <371E62CC.D66845F1@aspi.net>
References: <371d5c4c.2898786@news.io.com>
Newsgroups: sci.crypt
Lines: 170

Terry Ritter wrote:
>
> On Tue, 20 Apr 1999 18:50:48 -0700, in <371D2EF8.34DC6802@acm.org>, in
> sci.crypt Jim Gillogly <jim@acm.org> wrote:
>
> >I think Terry Ritter's right to be concerned about having essentially
> >everyone move to a single new cipher. If the danger isn't obvious,
> >consider the analogy with biological systems, where a species with no
> >genetic diversity can be wiped out by a single virus incident. Or with
> >computer systems, where something like Melissa can cause widespread
> >annoyance and some down time because almost everyone is using the
> >same operating system and office software suite.
>
> I think I have a right to cheer at this agreement with my major point.
>
> >I also agree with him that a careful concatenation of ciphers can
> >help limit the damage.
>
> And then I cheer again at this agreement with part of my proposed
> solution package.
>
> >I think we may disagree on what kinds of
> >ciphers would be most appropriate as choices for concatenation,
> >since I prefer ciphers that good analysts have tried and failed to
> >break over ciphers that nobody with cryptanalytical experience has
> >looked at. I define a good analyst as someone who has broken a
> >difficult system.
>
> Then I assume you are willing to make the services of such an analyst
> available free of charge and without delay. The way it is now, one
> cannot get such analysis unless one is a particular type of person,
> working in a few selected environments, and with particular types of
> design. Having inherited a democracy, I am unwilling to give that up
> for supposed advantages which, in the limit, do not give us what we
> want anyway. I think people should be able to select their own
> ciphers based on any criteria they want, including superstition and
> innuendo.
>
> >However, I (like John Savard) think Terry overstates some issues.
> >Here's a case in point:
> >
> >Terry Ritter wrote:
> >> Your position, dare I state it, is that you *can* estimate the
> >> capabilities of your Opponents.
> >
> >In another article he wrote:
> >> But the only thing being "measured" here is the open, academic
> >> analysis. The *real* experts do not play this way. We thus have no
> >> way to understand their capabilities. The strength value measured on
> >> academics cannot apply to the real problem.

> >
> >These and similar remarks suggest that a conservative threat analysis
> >must regard the opponents as god-like in their cryptanalytic
> >capabilities.
>
> If that is what you take from these comments (in their proper
> context), I am not surprised that you call my position overstated.
> However, you have exaggerated my position.
>
> In particular, I doubt I have ever said the Opponents are "god-like."
> As far as I can recall, the only people I have accused of being
> "god-like" are the crypto gods who seem to be able to predict: 1) the
> future strength of a cipher, based on past tests; and 2) the
> capabilities of unknown Opponents, based on the capabilities of known
> academics.
>
> >Of course in the limit this isn't useful, since we
> >would have no more confidence in a concatenation of ciphers against
> >an opponent like this than we would in a single cipher.
>
> And so, clearly, I do not so assume. Since I do not assume that an
> Opponent has unlimited capabilities, this comment strongly
> misrepresents my arguments.
>
> But what *are* we to assume? Even a *modest* "value" for Opponent
> capabilities is also "not useful" to us. This is because it is
> (virtually) impossible to *measure* knowledge, experience, and
> innovation. And then it is impossible to *measure* cipher strength.
> So we first don't know the difficulty of the problem, and then don't
> know the capabilities our Opponents can bring to the solution. This
> naturally leave us in a quandary, even *without* assuming unlimited
> capabilities. The problem is *not* that we should assume reasonable
> value for Opponent capabilities, the problem is that *any* such values
> and their implications are unknown, uncalibrated, and unuseful.
>
> I suggest that this whole line of inquiry (into cipher strength and
> Opponent strength) is a waste of time. Since we know that
> single-cipher failures are possible, we can work to fix that. Since I
> assume the triple-cipher scheme will work, it is clear that I do not
> assume unlimited Opponent capabilities. I do assume that whatever
> capabilities they do have will be stressed far harder with
> multi-ciphering than single ciphering. I think this is a reasonable
> assumption.

Some clarification may be called for in that your statements can be
construed as claims that cipher diversity solves the problem of inferior
talent/resources/etcetera with respect to dark-side adversaries and
future adversaries of all shades. I believe this absolutist position to
be false.

Your statements can also be constrused to claim that cipher diversity
will reduce whatever gap exists. I believe this relative position to be
true.

>
> Moreover, by using a wide variety of ciphers, we act to limit the
> amount of data disclosed by any break that does occur. I do assume
> that this will reduce the attraction of cryptanalysis, by limiting the
> eventual payoff. Again, I think this a reasonable assumption.

Some consideration also has to be given to the definition of payoff.
The dark-side adversaries get payoff in reaching thei information
goals. But academic researchers get payoff by earning the admiration of
their peers. That admiration can be earned in the absence of sccessful
attacks on a cipher system. A successful attack on a component of a
cipher system would be just as admirable as a successful attack on a
homogeneous cipher. Thus the cipher collection is not immune to attack
by reason of its lack of information leakage. A large body of talented
attackers will still be just as motivated as they are now.

>
> >However, we do have ways to estimate the capabilities of the opponents.
> >I suggest that the government cryptologic agencies of the US and UK
> >represent conservative surrogates for the cryptological skills of the
> >strongest opponents, and we have seen several unclassified examples of
> >times when they were less than perfect.
> >
> >In one case (factoring circa 1973) the UK agency was no further
> >advanced than the academic community, and academic advances in that
> >field were made shortly thereafter. In two other cases the US agency
> >made embarrassingly public blunders (the Clipper checksum exploited
> >by Matt Blaze, and the SHA/SHA-1 botch that they noticed and fixed
> >themselves) that would not have been made if they were omniscient.
> >I don't include Biham's work suggesting SKIPJACK is not a conservative
> >design, since we don't know that it has to be -- for all we know, there
> >are wads of supporting theorems that it's precisely as strong as it needs
> >to be for its size. We do have a couple of other cases of classified
> >discoveries and corresponding unclassified ones: IBM's differential
> >cryptanalysis (15 years) and CESG's non-secret encryption (4 years).
> >There are also training exercises (the Zendian Problem and a British
> >special intelligence course) which anyone can use to compare their skills
> >with advanced cipher school students of the 1960s. The latter does not,
> >of course, give the peak strength of the best cryppies, but does suggest
> >a starting point for the curve. Finally, we have retired NSA cryppie
> >Robert H. Morris's remarks at Crypto '95, where he said that by the

> >middle to late 1960's cryptanalysis had become less cost-effective than
> >other methods of gaining the information. One may choose to disbelieve
> >him, but I don't.
> >
> >In any case, we do have some data points on the capabilities of the
> >strongest potential opponents, and assuming they're perfect would be
> >overly conservative.
>
> First, none of this tells us about the future. Yet all operation of a
> cipher takes place in the future, after that cipher is designed.
> Unless we have a reasonable way to predict future capabilities, we are
> necessarily forced into conservative measures.
>
> Next, I think it is dangerous to assume our Opponents are the
> intelligence services we know. In another message I suggested that if
> the problem was only NSA (the way it is now), we would not have much
> of a problem. But NSA is only an *example* of an Opponent, and not
> necessarily even the most advanced example in particular areas of the
> technology. We having intractable problems in making any serious
> extrapolations from this data. Again I suggest that this avenue is
> both unfruitful and dangerous.
>
> ---
> Terry Ritter ritter@io.com http://www.io.com/~ritter/
> Crypto Glossary http://www.io.com/~ritter/GLOSSARY.HTM

Subject: Re: Question on confidence derived from cryptanalysis.
Date: Wed, 21 Apr 1999 18:59:45 GMT
From: ritter@io.com (Terry Ritter)
Message-ID: <371e2012.6178031@news.io.com>
References: <371E62CC.D66845F1@aspi.net>
Newsgroups: sci.crypt
Lines: 109

On Wed, 21 Apr 1999 19:44:12 -0400, in <371E62CC.D66845F1@aspi.net>,
in sci.crypt "Trevor Jackson, III" <fullmoon@aspi.net> wrote:

>Terry Ritter wrote:
>[...]
>Some clarification may be called for in that your statements can be
>construed as claims that cipher diversity solves the problem of inferior
>talent/resources/etcetera with respect to dark-side adversaries and
>future adversaries of all shades. I believe this absolutist position to
>be false.

I am not sure that I have made such a claim, which I also think is
false.

I don't know what could be clearer than my repeated statement that we
can trust no cipher. Cipher use cannot build trust in strength.
Cryptanalysis cannot certify strength. We must always be aware that
failure is a possibility, and we are even unable to estimate that
probability. When the consequences of cipher failure are
catastrophic, we simply cannot afford to depend on any one cipher.

The many-cipher part of the fix package has multiple goals, the first
being to compartmentalize information so that if the cipher (which we
do not and can not trust!) protecting that information fails, we do
not lose everything, throughout all society.

An implicit part of using multiple ciphers is that we change ciphers
at various times, so that we personally or corporately have similar
protection (i.e., cipher failure exposes only part of our
information). Once we have a way to change ciphers quickly, we have
vastly reduced the consequences of an academic break which finds a
weakness in our cipher. If any of our ciphers are found wanting, we
just use something else. No big deal.

With respect to the talents of the "dark-side adversaries" (a view
with which I doubt they would agree), we certainly must assume that
they have far greater resources than we do. But even their vast
resources are not unlimited; they must make the same tradeoffs any
project makes. So if they eventually do succeed against some cipher,
they expect a payoff from that success. If there is just one cipher
throughout society, that payoff will be huge, but if many ciphers are
used, the payoff will be minor.

By injecting a constant flow of new ciphers into the mix we force the
"adversaries" to "keep up" if they wish to maintain whatever level of
success they have. Each new cipher must be identified, acquired,
analyzed, broken, and software and perhaps hardware constructed to
automate the break. Their alternative is that less and less
information flows under ciphers which they can break. As we often
have seen discussed, it is far easier (thus cheaper) to construct a
new cipher than it is to analyze that cipher. This advantage in
cipher diversity provides *some* benefit, even if some of the ciphers
are weak. This is hardly an absolutist position.

Now, each of these paragraphs have discussed one or two specific
problems being solved by the fix package. I doubt that I would say
that *all* problems would be fixed, since that would be the cipher
argument in another guise. We cannot know. But very substantial

problems *are* fixed, and for the first time we take the battle to the cryptanalytic "adversaries" and make them pay a price. The alternative is to sit back and wish and hope for cipher strength, because we sure cannot prove it or test it.

>Your statements can also be constrused to claim that cipher diversity
>will reduce whatever gap exists. I believe this relative position to be
>true.

I'm not quite sure what this means, but thanks!

>> Moreover, by using a wide variety of ciphers, we act to limit the
>> amount of data disclosed by any break that does occur. I do assume
>> that this will reduce the attraction of cryptanalysis, by limiting the
>> eventual payoff. Again, I think this a reasonable assumption.
>
>Some consideration also has to be given to the definition of payoff.
>The dark-side adversaries get payoff in reaching thei information
>goals. But academic researchers get payoff by earning the admiration of
>their peers. That admiration can be earned in the absence of sccessful
>attacks on a cipher system. A successful attack on a component of a
>cipher system would be just as admirable as a successful attack on a
>homogeneous cipher.

I suppose you mean a particular cipher -- a component in the multi-cipher system. Not *just* as admirable perhaps, but admirable nevertheless. OK.

>Thus the cipher collection is not immune to attack
>by reason of its lack of information leakage.

I would say that we cannot trust any cipher, and we cannot trust any cipher system, including the fix package applied to current methods. No cipher system can possibly be immune. If we could prove or build "immune," we wouldn't need all this stuff.

>A large body of talented
>attackers will still be just as motivated as they are now.

Which is great, right? We want all the cryptanalysis we can get. If a cipher fails, we just use something else.

Maybe I had some trouble following your reasoning here.

Terry Ritter ritter@io.com http://www.io.com/~ritter/
Crypto Glossary http://www.io.com/~ritter/GLOSSARY.HTM

Subject: Re: Question on confidence derived from cryptanalysis.
Date: Wed, 21 Apr 1999 19:30:38 -0400
From: "Trevor Jackson, III" <fullmoon@aspi.net>
Message-ID: <371E5F9E.EE63FAE1@aspi.net>
References: <371D2EF8.34DC6802@acm.org>
Newsgroups: sci.crypt
Lines: 85

Jim Gillogly wrote:
>
> I think Terry Ritter's right to be concerned about having essentially
> everyone move to a single new cipher. If the danger isn't obvious,
> consider the analogy with biological systems, where a species with no
> genetic diversity can be wiped out by a single virus incident. Or with
> computer systems, where something like Melissa can cause widespread
> annoyance and some down time because almost everyone is using the
> same operating system and office software suite.
>
> I also agree with him that a careful concatenation of ciphers can
> help limit the damage. I think we may disagree on what kinds of
> ciphers would be most appropriate as choices for concatenation,
> since I prefer ciphers that good analysts have tried and failed to
> break over ciphers that nobody with cryptanalytical experience has
> looked at. I define a good analyst as someone who has broken a
> difficult system.
>
> However, I (like John Savard) think Terry overstates some issues.
> Here's a case in point:
>
> Terry Ritter wrote:
> > Your position, dare I state it, is that you *can* estimate the
> > capabilities of your Opponents.
>
> In another article he wrote:
> > But the only thing being "measured" here is the open, academic
> > analysis. The *real* experts do not play this way. We thus have no
> > way to understand their capabilities. The strength value measured on
> > academics cannot apply to the real problem.
>
> These and similar remarks suggest that a conservative threat analysis
> must regard the opponents as god-like in their cryptanalytic
> capabilities. Of course in the limit this isn't useful, since we
> would have no more confidence in a concatenation of ciphers against

> an opponent like this than we would in a single cipher.
>
> However, we do have ways to estimate the capabilities of the opponents.
> I suggest that the government cryptologic agencies of the US and UK
> represent conservative surrogates for the cryptological skills of the
> strongest opponents, and we have seen several unclassified examples of
> times when they were less than perfect.
>
> In one case (factoring circa 1973) the UK agency was no further
> advanced than the academic community, and academic advances in that
> field were made shortly thereafter. In two other cases the US agency
> made embarrassingly public blunders (the Clipper checksum exploited
> by Matt Blaze, and the SHA/SHA-1 botch that they noticed and fixed
> themselves) that would not have been made if they were omniscient.
> I don't include Biham's work suggesting SKIPJACK is not a conservative
> design, since we don't know that it has to be -- for all we know, there
> are wads of supporting theorems that it's precisely as strong as it needs
> to be for its size. We do have a couple of other cases of classified
> discoveries and corresponding unclassified ones: IBM's differential
> cryptanalysis (15 years) and CESG's non-secret encryption (4 years).
> There are also training exercises (the Zendian Problem and a British
> special intelligence course) which anyone can use to compare their skills
> with advanced cipher school students of the 1960s. The latter does not,
> of course, give the peak strength of the best cryppies, but does suggest
> a starting point for the curve. Finally, we have retired NSA cryppie
> Robert H. Morris's remarks at Crypto '95, where he said that by the
> middle to late 1960's cryptanalysis had become less cost-effective than
> other methods of gaining the information. One may choose to disbelieve
> him, but I don't.
>
> In any case, we do have some data points on the capabilities of the
> strongest potential opponents, and assuming they're perfect would be
> overly conservative.

There's no need to assume perfection or god-like omniscience to motivate as conservative an approach as possible. Considerations of our own ignorance regarding advances to be made in the open community during the life cycles of information we want to potect with today's tools dwarfs any sensible interpretation of current adversarial strength.

And with respect to adversaries from the dark side, the failures you mentioned do indicate that they err and thus are human. But they will always be at least as strong as the open community. We have no real clue hom much stronger they actually are, or _will be_.

Note also that Morris's statement is a relative statement. It can be construed to mean that cryptanalysis is less effective than before, or that "other methods" have become so much more effective that the _relative_ worth of crypto is less. The absolute worth could still be quite high and his statement could still be valid.

Subject: Re: Question on confidence derived from cryptanalysis.
Date: Tue, 20 Apr 1999 16:20:31 -0400
From: "Trevor Jackson, III" <fullmoon@aspi.net>
Message-ID: <371CE18F.30553812@aspi.net>
References: <371C025E.6AD4BAB8@raas.co.nz>
<371C58CF.286794F7@aspi.net>
Newsgroups: sci.crypt
Lines: 150

Geoff Thorpe wrote:
>
> Now, statement (1) is wrong. Maybe you cannot make estimates, and maybe
> you do not like the estimates others may employ. But there are ways to
> make estimates whose rationalisation is acceptable to those involved.
> That includes me. You also referred in that post to a complete lack of
> evidence but I think you yourself would be well positioned to refute
> that. Take every damned cipher you ever heard of (with any degree of
> cryptanalysis against it), exercise some personal judgement as to some
> measure of time+effort that the cipher was subjected to (by publishing
> authors - obviously not the spooks) before it became widely regarded as
> unacceptable, and take a look at the resulting distribution. That may
> not be a precise science, and of course it involves warm-fuzzy personal
> interpretations (time+effort) but it is not unacceptable for many
> people, particularly those who would otherwise be rendered with NO
> effective way to evaluate. I dare say that your distribution, if you've
> made semi-reasonable interpretations along the way, will show that a
> ciphers that lasted 10 years had a much better chance of lasting another
> year than the average "expected life". It's a very basic and common
> mathematical model/argument, and it's common sense.

An interesting concept. But in the absence of evidence I believe it to be wrong. Other than the obvious infant mortality due to negligence on the part of many cipher designers, I'd bet predictions described above are not very strong.

My reasonaing is that as a cipher gains the confidence of the community by surviving the gauntlet of previously known and/or straightforward attacks it will emerge from the background of many such ciphers and be attacked in a more serious way. The degree of emminence attracting more and more attention indicates that the share of offensive effort aimed at the cipher will continue to grow.

Given that there are few ciphers that have survived "the gauntlet" for a respectable period of time compared to the many ciphers without that maturity, the odds look to me much as Ritter described them. If I pick a young cipher, it may be broken tomorrow. If I pick an elderly cipher it may be broken tomorrow.

The appropriate metric for this kind of confidence is the expected wait for failure. Your claim amounts to the statement that the expected wait is longer for an elderly cipher than young one. I'm not comfortable with that.

It is an interesting thesis for study. Perhaps one of the frequent crypto students mght pick it up, define the criteria for judgement, and produce the results of the historical survey you suggested. Then we could discuss the merits of a carefully defined (tailoed?) set of criteria.

>
> I've already explained why I think that (2) is wrong - nobody knows any
> of this stuff FOR SURE, but you make a call when you don't have perfect
> information. Our Opponents are just well-paid versions of us, most of
> whom probably grew up around us, and who find their occupations not too
> unfathomably unethical to suffer day by day. I still maintain that what
> we can do and achieve is a statistical, probabilistic, and "confidence"
> variable that does not run along independantly of theirs. Depends how
> much George Orwell you read though ...
>
> > like to play and write papers. "They" just want information, and I'm
> > guessing just do whatever they got to do to get it - and searching
> > endlessly for little theoretical weaknesses is probably not their top
> > priority. That's not to say they don't do it and do it very well, but I
> > doubt their considerable advantages in resources are put so much to this
> > task as to make our abilities so incomparable or unrelated as some might
> > believe.
> >
> > A good point. However, we canot deal with their (secret) intentions,
> > but must anticipate their possible (even more secret) capabilities.
> > Thus amplifying the threat model is a sensible thing to do. It
> > eliminates some of the risk of catastrophically underestimating them by
> > enhancing the risk of expensively overestimating them.
>
> Sure thing - but the whole system does not collapse down to a binary
> system of "broken" and "not-broken-yet" ... as you say, you put together
> a threat model ... consistent with your requirements and using a chosen
> method for judging a components "worth", and amplify it here and there
> as appropriate. A lot like putting together a cost-proposal I guess ...
> add in your known prices, choose an acceptable value for the "unknowns",
> amplify the costs of all the "risky" bits, add x% profit on top - and
> then bang another 30% on top for good measure, and generally covering
> your butt some more.

A good model for planning the construction of a bridge or any major project. But I can;t apply this model to the construction of a security system because I have no clue how to make something 30% stronger. Do you?

>
> > It appears to me that he *does* agree (tho he can certainly speak for
> > himself), which is why he has repeatedly proposed the use of multiple
> > ciphers both to spread eggs across baskets, and to provide layered
> > security where warranted.
>
> 3 ciphers strung in a line is, to me, a cipher. You need all three in
> the same place and in the same order to have anything other than a
> "noise generator". Breaking 3 ciphers should be no more difficult than
> breaking one well designed one using 3 different stages

Here we part company. Your statement assumes that we will choose three ciphers whse strength adds up to the strength of one cipher that is well designed and uses several stages. But this is not the situation we face.

The situation we face is that we have dozens of reasonably strong ciphers, whose relative strengths are immeasurable. We may theorize about their relative merits, but we can't measure their strength in any fundamental sense. Given this I believe it makes sense to reduce risk of catastrophic failure by using a composite of the strongest ciphers of which we are aware.

In the limit (reductio...), we'd use all of the ciphers that we believe to be independent. The resulting monster would have severe implementation problems, but these are engineering issues that *can* be measured and shown solved. The creation of such a monster would be a never ending task because new ciphers worthy of inclusion are continually developed. This approach would limit the risk of catastrophic failure to the absolute minimum.

Now the monster described aboce is a silly example, but it is not at all absurd. This is why I believe Ritter's concept of composite ciphers has real value.

(if a cipher is
> based on one "idea", "primitive", or whatever then your vulnerability
> must surely be higher than distinct ideas employed serially?). It seems
> the argument put forth was more one of splitting the traffic

> (conceptually across time and application, not packet by packet I
> assume) across ciphers, and rotating the old out and the new in on a
> regular basis. I see this as unacceptable in a real-world scenario for
> reasons of interoperability & standardisation, as well as security.

No. Absolutely not. We already have tools that support multiple ciphers. We know how to extend and manage that process. It is not perfect, but it is "merely" engineering effort. Thus there are existential proofs that the interoperability and standardization issues to which you refer are minor compared to the issues of catastrophic failure that is the anti-objective of all our security efforts.

The only additions required to implement dynamic substitution of ciphers is a minor refinment of the existing cipher management capabilities. Automating what users can already do manually does not create the sort of problems that will defeat modern engineering, and eos not compromise security in the least. This issue is a red herring.

>
> Cheers,
> Geoff

Subject: Re: Question on confidence derived from cryptanalysis.
Date: Sun, 25 Apr 1999 11:31:58 +0200
From: "H. Ellenberger" <hansell@smile.ch>
Message-ID: <3722E10E.E9D76CC1@smile.ch>
References: <371C025E.6AD4BAB8@raas.co.nz>
<371C58CF.286794F7@aspi.net>
Newsgroups: sci.crypt
Lines: 95

Geoff Thorpe wrote:

> [...] 3 ciphers strung in a line is, to me, a cipher. You need all three in
> the same place and in the same order to have anything other than a
> "noise generator".

Assuming that cipher should mean strong cipher, then you are wrong:

Combine a very strong encription algorithm with it's decryption algorithm plus a
rot_13 and your result is in fact a weak rot_13!
This silly combination shows the importance of understanding how the _combined_
algorithms perform and is an argument towards your position.

> Breaking 3 ciphers should be no more difficult than
> breaking one well designed one using 3 different stages (if a cipher is
> based on one "idea", "primitive", or whatever then your vulnerability
> must surely be higher than distinct ideas employed serially?). [...]

The problem is that nobody exactly knows what algorithm results in a 'well designed' cipher.

Single algorithms combined to become a cipher are not necessarily strong by themselves, the stregth somehow results from the intelligent combination.

We can prove a bad design by demonstrating how to break it, but todays lack of a successful attack does not prove it's security.

However, there still is a valid argument to use multiple algorithms:
Since no algorithm is proved to be unbreakable, take those 10 beleived to be the most secure.
Build 5 pairs of them and verify you do not have a weak combination.
Now use one of the 5 pairs to encrypt a block, then annother pair for next block.

Alhough I can not prove that this apporach is stronger than a single 'well designed' algorithm, I am convinced that this approach is better.

a) Unless I had bad luck in selecting a pair, the combination of strong algorithms should be at least as safe as the single (strong) algorithm.
b) Should someone break a pair, he still can decrypt only 20% of my traffic. This strategy is comparable to stock investors spreading risks...

> Can I take apart a modern cipher and say "that's not a cipher - look,
> it's lots of little ciphers"?

I agree that you can, and I think that stregth results from two mechanisms:

a) Combination of basic principles in such a way that attack
in the middle becomes suficiently difficult.
b) If sufficiently varied ideas are combined, a new method
to break one of them may have less chance to be sucessfully
applied to the other.

> All I said was the division for me between
> 3 ciphers strung in a line and one cipher with 3 stages to it seems to
> be a question of packaging and patents.

Wrong.

My personal conclusion:

If science could _prove_ that a certain feasible algorithm
is unbreakable, everybody could use it and there would be no
need for annother one (and probably this algorithm would be
a intelligent combination of various principles).

Without such a proved algorithm it is indeed too risky to
use a single algorithm.
It is exposed to focused attacks from all sides, and in case
it sould break, damages are too important.

> [..] I still think "fixed", or at least "slow moving") collection of
> ciphers jumbled up to disperse the impact a break in any one
> configuration would have.

That's the way to go. In case one of the ciphers shows to be
less secure, it can be instantly removed from the collection
without disruption of all secure communication that would
occur if only _one_ algorithm would be available in the
myriads of computers around the globe.

Subject: Re: Question on confidence derived from cryptanalysis.
Date: Sun, 25 Apr 1999 18:57:46 -0400
From: Geoff Thorpe <geoff@raas.co.nz>
Message-ID: <37239DEA.D0F677FD@raas.co.nz>
References: <3722E10E.E9D76CC1@smile.ch>
Newsgroups: sci.crypt
Lines: 90

Hi there,

"H. Ellenberger" wrote:
> > [...] 3 ciphers strung in a line is, to me, a cipher. You need all three in
> > the same place and in the same order to have anything other than a
> > "noise generator".
>
> Assuming that cipher should mean strong cipher, then you are
> wrong:
>
> Combine a very strong encription algorithm with it's
> decryption algorithm plus a
> rot_13 and your result is in fact a weak rot_13!
> This silly combination shows the importance of understanding
> how the _combined_
> algorithms perform and is an argument towards your position.

Of course, you are in fact agreeing with some of my reservations. I
don't know if you were following the thread but I was just saying that
certain established and unbroken (despite plentiful attempts) ciphers
are worthy of more "trust", "strength", "tested strength", "confidence"
or whatever than new-toy ciphers. I was saying this to disagree with
someone who viewed all not-yet-broken ciphers as equally "(in)secure"
and that we should utilise any/all ciphers we want in a variety of ways
to spread our risk. I don't completely disagree with the latter point as
much as the former.

But you are right ... sticking 2 toys in a line does not make a better
wall than triple-DES ... such measures, in my mind, have a bigger risk
of falsely convincing us of "security" than our potential opponents. As
you pointed out with your ROT-13 illustration ... if the ciphers exhibit
any relationship in behaviour then there's even a risk that the
combination is weaker than the components. No argument here.

My point was simply that regarding a cipher as an atomic algorithmic
unit seemed a narrow point of view - if I design two simple ciphers and
am of the opinion that joining them in some suitable way gives better
overall properties, can I not call the result a cipher? I am not however
a cipher designer, I will not do this, and don't really want to make
this point the central issue as there are others better suited to get
into it than I.

> > Breaking 3 ciphers should be no more difficult than
> > breaking one well designed one using 3 different stages (if a cipher is
> > based on one "idea", "primitive", or whatever then your vulnerability
> > must surely be higher than distinct ideas employed serially?). [...]
>
> The problem is that nobody exactly knows what algorithm
> results in a 'well
> designed' cipher.
>

> Single algorithms combined to become a cipher are not
> necessarily strong by themselves, the stregth somehow
> results from the intelligent combination.

I agree - but the intelligent combination could be termed a "cipher" yes? If so, deployment of that "cipher" is easier than deployment of 10 different ciphers, all with various modes of operation (and hence support in their implementations), and one "cipher" is easier to standardise on and get interoperable systems out of. It does however limit the distribution of patents and recognition for cipher designers.

> We can prove a bad design by demonstrating how to break it,
> but todays lack of a
> successful attack does not prove it's security.

No, but if you think triple-DES is as "secure", or rather deserves the same "confidence", as mysticTarot128 then you agree with Terry, otherwise you agree with me. Or you may have no opinion on that. I think the chances that something like triple-DES will fall in the next, say, year is less than the chances for some new toy. Terry thinks that is a naive view and "un-scientific".

> However, there still is a valid argument to use multiple
> algorithms:
> Since no algorithm is proved to be unbreakable, take those
> 10 beleived to be the
> most secure.

Not possible according to the views I was disagreeing with. Apparently we must employ any and all ciphers we like and intermingle them in some kind of "risk-dispertion" technique that uses combinations, layers, etc. I see the conservative and simplified application of that technique as having some possible merits (if the technique itself is not subject to an all-or-nothing break), but I still hold to the concept that "less is more", particularly when we have seen before that obscurity doesn't really hide security, it hides weaknesses.

Regards,
Geoff

Subject: Re: Question on confidence derived from cryptanalysis.
Date: Mon, 19 Apr 1999 06:10:27 GMT
From: ritter@io.com (Terry Ritter)
Message-ID: <371ac7ce.40448497@news.io.com>
References: <371A6145.FAE5E8B@raas.co.nz>
<3718e5e9.9093614@news.io.com>
Newsgroups: sci.crypt
Lines: 78

On Sun, 18 Apr 1999 18:48:37 -0400, in <371A6145.FAE5E8B@raas.co.nz>, in sci.crypt Geoff Thorpe <geoff@raas.co.nz> wrote:

I spent some hours responding to this huge article, and only at the end realized (I assume correctly) that most of the controversy was about something which I consider peripheral to the main issue. So I am going to separate that other stuff off and ignore it so we don't loose sight of the forest. This is not intended to disrespect the effort in the original posting -- as I said, I have already made comparable effort in the response you do not see. But nobody wants to read huge postings, and all the points get lost anyway.

>Hello,
>
>Terry Ritter wrote:
>> >You want to sound a cautionary note that we all risk being naive and
>> >over-confident in our "cryptanalytic testing" of ciphers - excellent
>> >point and it is well taken.
>>
>> No, the point is NOT well-taken. It is ignored and brushed off as
>> trivial and known. Then everyone sticks their head in the sand again
>> until I bring it up again. This has happened for years.
>
>Once again, we are in disagreement - philosophically and factually it
>would appear. From your postings, I can understand why you think this,
>but it is based on a premise I simply do not accept and will not no
>matter how many times you repeat it. Namely, that repeated cryptanalytic
>testing does not provide a measure of the tested strength of a cipher.

OK, that is the peripheral cul-de-sac. I believe it, and believe it can be successfully argued, but it is a side-issue nevertheless.

My main argument starts out that no matter how much analysis is done, there is always the *possibility* that a cipher may fail anyway. I doubt anyone disagrees with this.

Since cipher failure is *possible*, we need to look at the consequences of failure: If this is to be the one standard cipher for society, the results of such failure would be *catastrophic*. Again, hardly controversial stuff.

We can do something about this: We can innovate various procedures and protocols to avoid single-cipher failure. As a general concept,

it is hard to imagine that even *this* is controversial. The actual technical details, of course, are arguable and changeable.

The package I have proposed includes compartmentalizing our data under different ciphers, thus reducing the amount of data at risk of any single cipher failure. (This virtually requires us to change ciphers frequently.) I also proposed multi-ciphering as a matter of course (reducing the likelihood of failure), and having a growing body of ciphers from which to choose. Other proposals can of course be made.

At this point, I see arguments for doing nothing (if the fix is too costly compared to the value at risk) and that the fix is worse than the original problem. The first depends upon the extent of the value at risk, which of course will include financial data, so the risk will be very, very high, without much argument. The probability of failure is the cul-de-sac argument itself, and may be hard to resolve. But even a very low probability may not be acceptable; it would not be acceptable to me.

The second part arguments are technical, but we can include the best-tested cipher in the multi-cipher stack. In this case, I think most would agree that -- properly done -- the overall strength could not be weaker than the tested cipher. And I think most would agree that this would indeed help prevent the single-point cipher failure which (almost) everyone will admit is at least *possible*.

Really, after going though this stuff at great length, I don't see much controversy here. No fireworks tonight: Sorry.

Terry Ritter ritter@io.com http://www.io.com/~ritter/
Crypto Glossary http://www.io.com/~ritter/GLOSSARY.HTM

Subject: Re: Question on confidence derived from cryptanalysis.
Date: Mon, 19 Apr 1999 21:48:09 -0400
From: "Trevor Jackson, III" <fullmoon@aspi.net>
Message-ID: <371BDCD9.673B1A34@aspi.net>
References: <371ac7ce.40448497@news.io.com>
Newsgroups: sci.crypt
Lines: 123

I rise (type) to speak in support of the proposition. There are two concepts from the field of economics that bear on this issue. The first concept is marginal utility as applied to the efforts of the crypto community. The second is the tendecy of the market to focus on a single "best" product.

The marginal utility concept applies to a cruptologist of any level of education/training/experience. It indicates that the value of an additional increment to the effort invested so far in an "established" cipher may be much less than tha value of that same effort applied to a neglected cipher. The economic case is that a rich man's millionth dollar (hundered billionth in Gates case) is worth less to the rich man than the poor man's thousandth dollar is worth to the poor man. This principle is absolutely fundamental to any economic analysis.

It is also fundamental and valuable in the field of operations reseach in that the fundamental, and mostly provable, assumption is that there is a "happy mix" that dominates all other mixtures of resource deployments. That looking for, and focusing on, "the" critical resource is too simplistic.

IMHO, diversity is a truly excellent thing in crypto. The field is young. There is *lots* of room for innovation.

The tendency of the market to focus on a single (or few) best product(s) is well established. The true operational basis for this is most often simple laziness. The theoretical basis is that concentrated effort will produce a better best than that same effort spread over a wide variety of options. If one company can dominate a market it can achieve economies of scale in production/design/etcetera.

The narrowing of the market is often seen in "industry shakeouts" where a developing industry with lots of vendors ranging from garage scale to fortune 10,000 scale merge/aquire/fail producing a "mature" market. Most consumer/customers actually like the simplified option menu of the mature market because fewer evaluations are necessary (laziness) and the risk of a really bad choice has been eliminated because the minimum and average quality of products in a mature market are usually much higher than those of a widely diverse market.

IMHO, this tendency should be resisted because I believe that cipher design does not benfit from economy of scale while cipher analysis certainly does.

Two concepts for $0.2. (special discount today only)

Terry Ritter wrote:
>
> On Sun, 18 Apr 1999 18:48:37 -0400, in <371A6145.FAE5E8B@raas.co.nz>,
> in sci.crypt Geoff Thorpe <geoff@raas.co.nz> wrote:
>

> I spent some hours responding to this huge article, and only at the
> end realized (I assume correctly) that most of the controversy was
> about something which I consider peripheral to the main issue. So I
> am going to separate that other stuff off and ignore it so we don't
> loose sight of the forest. This is not intended to disrespect the
> effort in the original posting -- as I said, I have already made
> comparable effort in the response you do not see. But nobody wants to
> read huge postings, and all the points get lost anyway.
>
> >Hello,
> >
> >Terry Ritter wrote:
> >> >You want to sound a cautionary note that we all risk being naive and
> >> >over-confident in our "cryptanalytic testing" of ciphers - excellent
> >> >point and it is well taken.
> >>
> >> No, the point is NOT well-taken. It is ignored and brushed off as
> >> trivial and known. Then everyone sticks their head in the sand again
> >> until I bring it up again. This has happened for years.
> >>
> >Once again, we are in disagreement - philosophically and factually it
> >would appear. From your postings, I can understand why you think this,
> >but it is based on a premise I simply do not accept and will not no
> >matter how many times you repeat it. Namely, that repeated cryptanalytic
> >testing does not provide a measure of the tested strength of a cipher.
>
> OK, that is the peripheral cul-de-sac. I believe it, and believe it
> can be successfully argued, but it is a side-issue nevertheless.
>
> My main argument starts out that no matter how much analysis is done,
> there is always the *possibility* that a cipher may fail anyway. I
> doubt anyone disagrees with this.
>
> Since cipher failure is *possible*, we need to look at the
> consequences of failure: If this is to be the one standard cipher for
> society, the results of such failure would be *catastrophic*. Again,
> hardly controversial stuff.
>
> We can do something about this: We can innovate various procedures
> and protocols to avoid single-cipher failure. As a general concept,
> it is hard to imagine that even *this* is controversial. The actual
> technical details, of course, are arguable and changeable.
>
> The package I have proposed includes compartmentalizing our data under
> different ciphers, thus reducing the amount of data at risk of any
> single cipher failure. (This virtually requires us to change ciphers
> frequently.) I also proposed multi-ciphering as a matter of course
> (reducing the likelihood of failure), and having a growing body of
> ciphers from which to choose. Other proposals can of course be made.
>
> At this point, I see arguments for doing nothing (if the fix is too
> costly compared to the value at risk) and that the fix is worse than
> the original problem. The first depends upon the extent of the value
> at risk, which of course will include financial data, so the risk will
> be very, very high, without much argument. The probability of failure
> is the cul-de-sac argument itself, and may be hard to resolve. But
> even a very low probability may not be acceptable; it would not be
> acceptable to me.
>
> The second part arguments are technical, but we can include the
> best-tested cipher in the multi-cipher stack. In this case, I think
> most would agree that -- properly done -- the overall strength could
> not be weaker than the tested cipher. And I think most would agree
> that this would indeed help prevent the single-point cipher failure
> which (almost) everyone will admit is at least *possible*.
>
> Really, after going though this stuff at great length, I don't see
> much controversy here. No fireworks tonight: Sorry.
>
> ---
> Terry Ritter ritter@io.com http://www.io.com/~ritter/
> Crypto Glossary http://www.io.com/~ritter/GLOSSARY.HTM

Subject: Re: Question on confidence derived from cryptanalysis.
Date: Mon, 19 Apr 1999 15:03:34 GMT
From: aquiranx@goliatx.ugr.es (Gurripato (x=nospam))
Message-ID: <371b4488.28191695@news.cica.es>
References: <371BDCD9.673B1A34@aspi.net>
Newsgroups: sci.crypt
Lines: 25

On Mon, 19 Apr 1999 21:48:09 -0400, "Trevor Jackson, III"
<fullmoon@aspi.net> wrote:

>The tendency of the market to focus on a single (or few) best product(s)
>is well established. The true operational basis for this is most often
>simple laziness. The theoretical basis is that concentrated effort will
>produce a better best than that same effort spread over a wide variety
>of options. If one company can dominate a market it can achieve
>economies of scale in production/design/etcetera.

I disagree. There must be some reason why a product is well

established, not necessarily quality. The VCR-format war of the 80´s
was won by the VHS system (in the sense that they sell more than any
other system). But it is well-known that Beta offers higher quailty.
VHS won mainly for marketing problems: Sony kept the Beta patents for
himself, while everyone else went to VSH. And if we talk about OS,
Windows95 (best seller the world over) falls miserably in quality
aspects (hangouts, crashes, etc) to others like Linux or MacOS.

I recall an article on Scientific American about things like
railway width being accepted by the market not for its quality, but
rather on a sort of chaos-like process. Sorry, I don´t remember the
SA issue.

Subject: Re: Question on confidence derived from cryptanalysis.
Date: Tue, 20 Apr 1999 01:18:00 -0400
From: "Trevor Jackson, III" <fullmoon@aspi.net>
Message-ID: <371C0E08.8D00868A@aspi.net>
References: <371b4488.28191695@news.cica.es>
Newsgroups: sci.crypt
Lines: 30

Gurripato (x=nospam) wrote:
>
> On Mon, 19 Apr 1999 21:48:09 -0400, "Trevor Jackson, III"
> <fullmoon@aspi.net> wrote:
>
> >The tendency of the market to focus on a single (or few) best product(s)
> >is well established. The true operational basis for this is most often
> >simple laziness. The theoretical basis is that concentrated effort will
> >produce a better best than that same effort spread over a wide variety
> >of options. If one company can dominate a market it can achieve
> >economies of scale in production/design/etcetera.
>
> I disagree. There must be some reason why a product is well
> established, not necessarily quality. The VCR-format war of the 80´s
> was won by the VHS system (in the sense that they sell more than any
> other system). But it is well-known that Beta offers higher quailty.
> VHS won mainly for marketing problems: Sony kept the Beta patents for
> himself, while everyone else went to VSH. And if we talk about OS,
> Windows95 (best seller the world over) falls miserably in quality
> aspects (hangouts, crashes, etc) to others like Linux or MacOS.
>
> I recall an article on Scientific American about things like
> railway width being accepted by the market not for its quality, but
> rather on a sort of chaos-like process. Sorry, I don´t remember the
> SA issue.

You are certainly free to disagree with the marketing theory that says
mature markets are better. After all, I disagree with it too.

I find it especially unsuitable for the field of crypto.

Terry Ritter, his *current address*, and his *top page*.

Last updated: 1999-05-12

Block Cipher Modes for One-Block Messages?

A *Ciphers By Ritter* Page

If we use a block cipher, how can we avoid causing the message to expand when ciphering? And what do we do when the message is shorter than a block?

Contents

- 1999-01-04 Jon Becker: "I have a question about Ciphertext Stealing...." "...how does one use CTS to generate the ciphertext for a plaintext message which is less than the cipher's block size...."
- 1999-01-04 John Savard: "...no technique that sufficiently resembles 'ciphertext stealing' to be called a case of it is applicable to such short messages."
- 1999-01-04 Jon Becker: "According to RFC 2040: This mode handles any length of plaintext and produces ciphertext whose length matches the plaintext length."
- 1999-01-04 Frank Gifford: "If you want to encrypt a message which has a smaller blocksize than your cipher, you can tack on 0 bits to the beginning, encrypt that, chop off the first several bits and send the answer."
- 1999-01-05 dscott@networkusa.net: "Actuall the above does not work since getting all zeros does not even necessiarly mean you have the right soultion and more than one solution can exist."
- 1999-01-05 John Savard: "...'any length' may simply have meant any length that may arise in practice...."
- 1999-01-04 dscott@networkusa.net: "...files that are too short can't be encrypted at all unless one uses the option of adding random padding...."

```
Subject: CTS a la Schneier, Rivest
Date: 4 Jan 1999 19:16:49 GMT
From: becker@Xenon.Stanford.EDU (Jon Becker)
Message-ID: &LT;76r431$h5g$1@nntp.Stanford.EDU>
Newsgroups: sci.crypt
Lines: 31


I have a question about Ciphertext Stealing, particularly as it
relates to RC5.  In "Applied Cryptography" (Schneier, p. 195) as well
as in RFC 2040 (section 8, on RC5-CTS), CTS is described as a way of
using a block cipher to encrypt a message of any size and have the
ciphertext length match the plaintext length.

In both documents, however, the descriptions assume the existence of a
second-to-last block of full size.  My question is, how does one use
CTS to generate the ciphertext for a plaintext message which is less
than the cipher's block size (so that the aforementioned assumption
doesn't hold)?

For example, if I'm using RC5 with a 64-bit block size and I have a
plaintext message which is 6 bytes long, what will the ciphertext be
using RC5-CTS?

I suppose it would be possible to act as if the IV is the encryption
of the second-to-last plaintext block.  Then it would be XORed with
the plaintext padded with 0s and encrypted to form the second-to-last
ciphertext block.  This would be transmitted as a new IV, while the
first 6 bytes of the original IV would be transmitted as the real
ciphertext.

I'm reasonably sure this method maintains security, but it does have
the disadvantage of changing the IV before and after encryption, which
is pretty much contrary to the idea of an IV.

Can anyone point me to some definitive documentation on the subject?

-Jon
```

```
Subject: Re: CTS a la Schneier, Rivest
Date: Mon, 04 Jan 1999 22:59:39 GMT
From: jsavard@tenMAPSONeerf.edmonton.ab.ca (John Savard)
Message-ID: &LT;36914690.3976436@news.prosurfr.com>
References: &LT;76r431$h5g$1@nntp.Stanford.EDU>
Newsgroups: sci.crypt
Lines: 34

becker@Xenon.Stanford.EDU (Jon Becker) wrote, in part:

>In both documents, however, the descriptions assume the existence of a
>second-to-last block of full size.  My question is, how does one use
>CTS to generate the ciphertext for a plaintext message which is less
>than the cipher's block size (so that the aforementioned assumption
>doesn't hold)?

There is no method to encipher a message of less than the blocksize in
length, using ECB mode, to produce an enciphered message of the same
length.

Thus, while one could still use the block cipher, either by padding
the message to a full block, or by using a stream cipher mode such as
counter mode, no technique that sufficiently resembles "ciphertext
stealing" to be called a case of it is applicable to such short
messages.

Essentially, the only difference between the complicated-looking
"ciphertext stealing" technique depicted in AC and simply enciphering
each complete block of the message, and then, if an incomplete block
is left unenciphered, enciphering the last 64 bits (or whatever the
blocksize is) of the message is that the ciphertext stealing technique
avoids alignment problems.

(Myself, I would tend to encipher the last 8 bytes - and use the
ciphertext stealing technique only to fill out the last byte if
incomplete.)

Clearly, you can't encipher the 'first 64 bits' and the 'last 64 bits'
of something less than 64 bits long.

John Savard
http://www.freenet.edmonton.ab.ca/~jsavard/index.html
```

```
Subject: Re: CTS a la Schneier, Rivest
Date: 4 Jan 1999 23:23:55 GMT
From: becker@Xenon.Stanford.EDU (Jon Becker)
Message-ID: &LT;76riib$mq8$1@nntp.Stanford.EDU>
References: &LT;36914690.3976436@news.prosurfr.com>
Newsgroups: sci.crypt
Lines: 42

In article &LT;36914690.3976436@news.prosurfr.com>,
John Savard &LT;jsavard@tenMAPSONeerf.edmonton.ab.ca> wrote:
>becker@Xenon.Stanford.EDU (Jon Becker) wrote, in part:
>
>>In both documents, however, the descriptions assume the existence of a
>>second-to-last block of full size.  My question is, how does one use
>>CTS to generate the ciphertext for a plaintext message which is less
>>than the cipher's block size (so that the aforementioned assumption
>>doesn't hold)?
>
>There is no method to encipher a message of less than the blocksize in
>length, using ECB mode, to produce an enciphered message of the same
>length.
>
>Thus, while one could still use the block cipher, either by padding
>the message to a full block, or by using a stream cipher mode such as
>counter mode, no technique that sufficiently resembles "ciphertext
>stealing" to be called a case of it is applicable to such short
>messages.
>
>Essentially, the only difference between the complicated-looking
>"ciphertext stealing" technique depicted in AC and simply enciphering
>each complete block of the message, and then, if an incomplete block
>is left unenciphered, enciphering the last 64 bits (or whatever the
>blocksize is) of the message is that the ciphertext stealing technique
>avoids alignment problems.
>
>(Myself, I would tend to encipher the last 8 bytes - and use the
>ciphertext stealing technique only to fill out the last byte if
>incomplete.)
>
>Clearly, you can't encipher the 'first 64 bits' and the 'last 64 bits'
>of something less than 64 bits long.

Yes, clearly.  So we have a conundrum.  According to RFC 2040:

   This mode handles any length of plaintext and produces
   ciphertext whose length matches the plaintext length.

So what is one to make of that?  Is it just a mistake?

-Jon
```

```
Subject: Re: CTS a la Schneier, Rivest
Date: 4 Jan 1999 18:41:46 -0500
From: giff@eng.us.uu.net (Frank Gifford)
Message-ID: &LT;76rjjq$mgl@trebuchet.eng.us.uu.net>
References: &LT;76riib$mq8$1@nntp.Stanford.EDU>
Newsgroups: sci.crypt
Lines: 35

In article &LT;76riib$mq8$1@nntp.Stanford.EDU>,
Jon Becker &LT;becker@Xenon.Stanford.EDU> wrote:
>In article &LT;36914690.3976436@news.prosurfr.com>,
>John Savard &LT;jsavard@tenMAPSONeerf.edmonton.ab.ca> wrote:
>>becker@Xenon.Stanford.EDU (Jon Becker) wrote, in part:
>>
>>>In both documents, however, the descriptions assume the existence of a
>>>second-to-last block of full size.  My question is, how does one use
>>>CTS to generate the ciphertext for a plaintext message which is less
>>>than the cipher's block size (so that the aforementioned assumption
>>>doesn't hold)?
>>
>>There is no method to encipher a message of less than the blocksize in
>>length, using ECB mode, to produce an enciphered message of the same
>>length.

Take the last complete encrypted block and encrypt that again.  Then do
an XOR with an equal number of bytes from that result and the final bytes
of the message.  There is a potential problem of the encryption of
two messages which differ only in those bytes, however, so this is more
of a "here's a way to do it" situation.  In many cases, padding is added
to bring the message to a block size.

If you want to encrypt a message which has a smaller blocksize than your
cipher, you can tack on 0 bits to the beginning, encrypt that, chop off
the first several bits and send the answer.  Your recipient then can try
decrypting all possible starting bit values and which have the sent message
as the remainder.  When he gets a decryption starting with all zero bits,
he knows he has the right answer.  [This example is a little far fetched,
I know - it's just for illustration.]

-Giff

--
giff@uu.net       Too busy for a .sig
```

```
Subject: Re: CTS a la Schneier, Rivest
Date: Tue, 05 Jan 1999 13:10:27 GMT
From: dscott@networkusa.net
Message-ID: &LT;76t303$jv$1@nnrp1.dejanews.com>
References: &LT;76rjjq$mgl@trebuchet.eng.us.uu.net>
Newsgroups: sci.crypt
Lines: 31

In article &LT;76rjjq$mgl@trebuchet.eng.us.uu.net>,
  giff@eng.us.uu.net (Frank Gifford) wrote:
snip ...

>
> If you want to encrypt a message which has a smaller blocksize than your
> cipher, you can tack on 0 bits to the beginning, encrypt that, chop off
> the first several bits and send the answer.  Your recipient then can try
> decrypting all possible starting bit values and which have the sent message
> as the remainder.  When he gets a decryption starting with all zero bits,
> he knows he has the right answer.  [This example is a little far fetched,
> I know - it's just for illustration.]

 Actuall the above does not work since getting all zeros does not
even necessiarly mean you have the right soultion and more than one
solution can exist. I guess it means you have not played much with
encryption. There are several modes of encryption that allow short
blocks and most are enferior. Even the mode in PGP 2.6.3 is worse than
ECB since it uses a stream generated by the key only then XOR s the
result with the file to be encrypted. This allows short blocks and
any lenght files. Many codes use this method. But it sucks.

David Scott
P.S. like at common chainning modes in the FAQ

--
http://cryptography.org/cgi-bin/crypto.cgi/Misc/scott19u.zip
http://members.xoom.com/ecil/index.htm

-----------== Posted via Deja News, The Discussion Network ==----------
http://www.dejanews.com/       Search, Read, Discuss, or Start Your Own
```

```
Subject: Re: CTS a la Schneier, Rivest
Date: 5 Jan 99 03:16:13 GMT
From: jsavard@ecn.ab.ca ()
Message-ID: &LT;369183fd.0@ecn.ab.ca>
References: &LT;76riib$mq8$1@nntp.Stanford.EDU>
Newsgroups: sci.crypt
Lines: 18

Jon Becker (becker@Xenon.Stanford.EDU) wrote:
: Yes, clearly.  So we have a conundrum.  According to RFC 2040:

:    This mode handles any length of plaintext and produces
:    ciphertext whose length matches the plaintext length.

: So what is one to make of that?  Is it just a mistake?

Yes. However, "any length" may simply have meant any length that may arise
in practice; the caveat concerning a message shorter than a single block
may have been a detail not worth the words it would have taken to mention
it: even in a standards document, people will sometimes speak without
using mathematical precision.

An oversight, or an oversimplification. Not even just a full-fledged
mistake, but less than a mistake.

John Savard
```

```
Subject: Re: CTS a la Schneier, Rivest
Date: Mon, 04 Jan 1999 23:00:17 GMT
From: dscott@networkusa.net
Message-ID: &LT;76rh61$nr4$1@nnrp1.dejanews.com>
References: &LT;76r431$h5g$1@nntp.Stanford.EDU>
Newsgroups: sci.crypt
Lines: 57

In article &LT;76r431$h5g$1@nntp.Stanford.EDU>,
  becker@Xenon.Stanford.EDU (Jon Becker) wrote:
>
> I have a question about Ciphertext Stealing, particularly as it
> relates to RC5.  In "Applied Cryptography" (Schneier, p. 195) as well
> as in RFC 2040 (section 8, on RC5-CTS), CTS is described as a way of
> using a block cipher to encrypt a message of any size and have the
> ciphertext length match the plaintext length.
>
> In both documents, however, the descriptions assume the existence of a
> second-to-last block of full size.  My question is, how does one use
> CTS to generate the ciphertext for a plaintext message which is less
> than the cipher's block size (so that the aforementioned assumption
> doesn't hold)?
>
> For example, if I'm using RC5 with a 64-bit block size and I have a
> plaintext message which is 6 bytes long, what will the ciphertext be
> using RC5-CTS?
>
> I suppose it would be possible to act as if the IV is the encryption
> of the second-to-last plaintext block.  Then it would be XORed with
> the plaintext padded with 0s and encrypted to form the second-to-last
> ciphertext block.  This would be transmitted as a new IV, while the
> first 6 bytes of the original IV would be transmitted as the real
> ciphertext.
>
> I'm reasonably sure this method maintains security, but it does have
> the disadvantage of changing the IV before and after encryption, which
> is pretty much contrary to the idea of an IV.
>
> Can anyone point me to some definitive documentation on the subject?
>
> -Jon
>


 Scott16u and scott19u also have this feature for long files the
length of the encrypted file can be exactly the same as the plaintext
file. However files that are too short can't be encrypted at all unless
one uses the option of adding random padding to the file which makes
the encrypted file longer but yet allows short plain text files to be
encrypted. It also has the advantage that short files that are identical
and encrypted with same key will be totally different in the encrypted
state. The method used realies on "wrapped PCBC" see source code and
it can be used with any block cipher it would greatly encrease the
strength of encryption compared to simple methods endorsed by Mr B.S
and his NSA friends.

 Just my opinion never having read his book.
David Scott

--
http://cryptography.org/cgi-bin/crypto.cgi/Misc/scott19u.zip
http://members.xoom.com/ecil/index.htm

-----------== Posted via Deja News, The Discussion Network ==----------
http://www.dejanews.com/       Search, Read, Discuss, or Start Your Own
```

Terry Ritter, his current address, and his top page.

Last updated: 1999-02-20

Random Access to Encrypted Data

A *Ciphers By Ritter* Page

Chaining modes are a virtual requirement for the secure use of ciphers with small blocks. But then how do we get random access?

Contents

- 1998-12-22 Marco Stolpe: "...I concluded that the only thing I can do is to use a block cipher with ECB mode."
- 1998-12-22 Bryan Olson: "You can divide the file into pages of, for example, two kilobytes, and encrypt these units with a better mode."
- 1998-12-22 neilmck@hotmail.com: "...place an 8-byte block containing a random number at the beginning of the file."
- 1998-12-22 dscott@networkusa.net: "One simple way to reduce your exposure would to be to use as large a block size as possible."
- 1998-12-23 Marco Stolpe: "Encrypting larger blocks of a file with CBC mode seems to be a good solution." "...the 4 bits of valuable information in one byte of plaintext are randomized by the upper 4 bits and that now one block of valuable(!) plaintext together with the 4 random bits encrypts to many different blocks of ciphertext."
- 1998-12-24 Terry Ritter: "I have been describing it on sci.crypt for the past few years as the 'homophonic' block cipher construction. It is 'homophonic' in that each different random value selects a different ciphertext for exactly the same data field."
- 1998-12-23 Kenneth Almquist: "Once you understand the theory behind CBC mode, coming up with an alternative which allows random access is straightforward."
- 1998-12-24 Terry Ritter: "...it can be a serious mistake to build a system which uses a file position in encryption."
- 1998-12-24 John Savard: "...one could use the XOR of a track and sector address for disk encryption without a problem."
- 1998-12-24 Terry Ritter: "...if there comes a day when the encrypted data is forgotten and the storage upgraded (which can be as simple as being away on a trip), the data will simply become usable."
- 1998-12-24 John Savard: "To conceal patterns in the plaintext, where you actually are restricted to ECB mode, you can always XOR each plaintext block with its address...."

```
Subject: Enhancement of EBC mode?
Date: Tue, 22 Dec 1998 09:13:31 +0100
From: Marco Stolpe &LT;marco.stolpe@usa.net>
Message-ID: &LT;367F54AB.173BBA95@usa.net>
Newsgroups: sci.crypt
Lines: 55

-----BEGIN PGP SIGNED MESSAGE-----
Hash: SHA1

Hello!

I'm planning to write an application where encryption of files
is needed but random access to the content of these files is
necessary.

- From chapter nine of the book "Applied Cryptography" by
Bruce Schneier (1996) I concluded that the only thing I can
do is to use a block cipher with ECB mode. In this book the
following negative aspects of ECB are mentioned:

" 1) Plaintext patterns are not concealed.
  2) Input to the block cipher is not randomized; it is the same
     as the plaintext.
  3) Plaintext is easy to manipulate; blocks can be removed,
     repeated, or interchanged. "

Although I don't know a good solution to 3), I considered the
following solution to 1) and 2):

- - Decrease the amount of information stored in each byte,
  for example by using the base64 encoding scheme.

- - Fill up the 'free' bits in each byte with random bits
  produced by a very good random sequence generator
  (for example the one proposed in chapter 17.14 of the
  book which I mentioned).

- - Encrypt the block with a normal block cipher.


Since I'm not a professional cryptographer, I would like to
know if such a procedure could increase security of ECB mode
in any way?

I'd like to hear from you,

Marco Stolpe

- --
PGP Key, ID 0x4F3FE0B5 should be available on a keyserver
Fingerprint:
D0AA F39C 0D9D 4AC8 D742  C0DB 3536 3D29 4F3F E0B5

-----BEGIN PGP SIGNATURE-----
Version: PGPfreeware 5.0i for non-commercial use
Charset: noconv

iQA/AwUBNn9UGjU2PSlPP+C1EQI3wwCgryDeEfLncaRIi0bXrYaNioGziG4An0aD
PsYMo4YqvIfmzvS+D4swjPoW
=stKf
-----END PGP SIGNATURE-----
```

```
Subject: Re: Enhancement of EBC mode?
Date: Tue, 22 Dec 1998 02:10:45 -0800
From: Bryan Olson &LT;bryan.olson@uptronics.com>
Message-ID: &LT;367F7025.3A767E97@uptronics.com>
References: &LT;367F54AB.173BBA95@usa.net>
Newsgroups: sci.crypt
Lines: 39

Marco Stolpe wrote:

> I'm planning to write an application where encryption of files
> is needed but random access to the content of these files is
> necessary.

[...]
> I concluded that the only thing I can
> do is to use a block cipher with ECB mode.

You actually have lots of options.  The most popular is
to work in units of pages.  You can divide the file into
pages of, for example, two kilobytes, and encrypt these
units with a better mode.  To randomly access byte 2567,
you'd need to decrypt the second page, and pull out the
byte at offset (2567-2048) within the page.

If you can afford an IV for each page, you can use one
of standard chaining modes.  If not, you can use a mode
that makes each ciphertext byte dependent on each byte
in the page, and the page number, and one IV for the
file.  See Peter Gutmann's SFS for an example.

    http://www.cs.auckland.ac.nz/~pgut001/

> - - Decrease the amount of information stored in each byte,
>   for example by using the base64 encoding scheme.
>
> - - Fill up the 'free' bits in each byte with random bits
>   produced by a very good random sequence generator
>   (for example the one proposed in chapter 17.14 of the
>   book which I mentioned).
>
> - - Encrypt the block with a normal block cipher.

Those are good ideas.  I think you'd still be better off
working in units larger than one typical cipher block.

--Bryan
```

```
Subject: Re: Enhancement of EBC mode?
Date: Tue, 22 Dec 1998 12:51:13 GMT
From: neilmck@hotmail.com
Message-ID: &LT;75o4k1$5q3$1@nnrp1.dejanews.com>
References: &LT;367F54AB.173BBA95@usa.net>
Newsgroups: sci.crypt
Lines: 27


> " 1) Plaintext patterns are not concealed.
>   2) Input to the block cipher is not randomized; it is the same
>      as the plaintext.
>   3) Plaintext is easy to manipulate; blocks can be removed,
>      repeated, or interchanged. "
>

Consider the following:

As for the plaintext patterns not being concealed, place an 8-byte block
containing a random number at the beginning of the file.  XOR this value with
your key before encrypting-decrypting.  Further XOR the key with the offset
of the data data block from the beginning of the file (the key is then unique
per file and per block).  (Note if you are using DEA and your key is spread
over 8-bytes you must be sure to XOR your offset with only key bits so that
adjacent blocks do not use the same key)

To detect manipulation of the data split your plaintext into 6-byte blocks and
add a two byte checksum to make an 8-byte block.  If the ciphertext is then
manipulated or blocks are switched within the file then after decryption the
checksum will not match the plaintext.



-----------== Posted via Deja News, The Discussion Network ==----------
http://www.dejanews.com/       Search, Read, Discuss, or Start Your Own
```

```
Subject: Re: Enhancement of EBC mode?
Date: Tue, 22 Dec 1998 13:44:58 GMT
From: dscott@networkusa.net
Message-ID: &LT;75o7oq$8d5$1@nnrp1.dejanews.com>
References: &LT;367F54AB.173BBA95@usa.net>
Newsgroups: sci.crypt
Lines: 92

In article &LT;367F54AB.173BBA95@usa.net>,
  Marco Stolpe &LT;marco.stolpe@usa.net> wrote:
> -----BEGIN PGP SIGNED MESSAGE-----
> Hash: SHA1
>
> Hello!
>
> I'm planning to write an application where encryption of files
> is needed but random access to the content of these files is
> necessary.
>
> - From chapter nine of the book "Applied Cryptography" by
> Bruce Schneier (1996) I concluded that the only thing I can
> do is to use a block cipher with ECB mode. In this book the
> following negative aspects of ECB are mentioned:
>
> " 1) Plaintext patterns are not concealed.
>   2) Input to the block cipher is not randomized; it is the same
>      as the plaintext.
>   3) Plaintext is easy to manipulate; blocks can be removed,
>      repeated, or interchanged. "
>
> Although I don't know a good solution to 3), I considered the
> following solution to 1) and 2):
>
> - - Decrease the amount of information stored in each byte,
>   for example by using the base64 encoding scheme.
>

   this would be a mistake since it would decrease the total
number of different types of blocks while increasing the number
of bloks that a hacker can look at. It would make plain text
attack easier.

> - - Fill up the 'free' bits in each byte with random bits
>   produced by a very good random sequence generator
>   (for example the one proposed in chapter 17.14 of the
>   book which I mentioned).

  It is very hard to get random data. Since you should assume
the attacker may have access to same random number generator.
This can also add more time to impliment in a good way than
you think.

>
> - - Encrypt the block with a normal block cipher.
>
> Since I'm not a professional cryptographer, I would like to
> know if such a procedure could increase security of ECB mode
> in any way?

 One simple way to reduce your exposure would to be to use
as large a block size as possible. And a simple way to protect
data in such an environment is to XOR the block number (index
number whatever) used in randomaccess fetch so that the bad
effects ECB mode are eliminated. Do this XOR before the block
is encrypted.
 The hardest part is finding a good method for the block size
you need. I will eventually write a method using 1675 bits of
key space for a scott8u method that would allow a varable block
size of 40 to 40,000 bits so those that have similar problems
can use it. You can write your own if you like just look at my
C code. for file encryption and instead of "file" use a "block"


>
> I'd like to hear from you,
>
> Marco Stolpe
>
> - --
> PGP Key, ID 0x4F3FE0B5 should be available on a keyserver
> Fingerprint:
> D0AA F39C 0D9D 4AC8 D742  C0DB 3536 3D29 4F3F E0B5
>
> -----BEGIN PGP SIGNATURE-----
> Version: PGPfreeware 5.0i for non-commercial use
> Charset: noconv
>
> iQA/AwUBNn9UGjU2PSlPP+C1EQI3wwCgryDeEfLncaRIi0bXrYaNioGziG4An0aD
> PsYMo4YqvIfmzvS+D4swjPoW
> =stKf
> -----END PGP SIGNATURE-----
>


--
http://cryptography.org/cgi-bin/crypto.cgi/Misc/scott19u.zip
http://members.xoom.com/ecil/index.htm

-----------== Posted via Deja News, The Discussion Network ==----------
http://www.dejanews.com/       Search, Read, Discuss, or Start Your Own
```

```
Subject: Re: Enhancement of EBC mode?
Date: Wed, 23 Dec 1998 16:18:50 +0100
From: Marco Stolpe &LT;marco.stolpe@usa.net>
Message-ID: &LT;368109DA.E70A98BD@usa.net>
References: &LT;75o7oq$8d5$1@nnrp1.dejanews.com>
Newsgroups: sci.crypt
Lines: 77

-----BEGIN PGP SIGNED MESSAGE-----
Hash: SHA1

Thanks to all who gave me an advice. Encrypting larger blocks
of a file with CBC mode seems to be a good solution.

But there's one statement which surprised me a little bit.
You (dscott@networkusa.net) wrote:

>> - - Decrease the amount of information stored in each byte,
>>     for example by using the base64 encoding scheme.
>>
> this would be a mistake since it would decrease the total
> number of different types of blocks while increasing the
> number of bloks that a hacker can look at. It would make
> plain text attack easier.

I think that base64 encoding is a bad example, especially
because it uses 65 instead of 64 (= 6 bits) different
characters.

The whole idea of mine was the following:

I've read the main danger of using ECB mode is that one block
of plaintext always encrypts to the same block of ciphertext.
That means that it is theoretically possible to create a
code book of plaintexts and corresponding ciphertexts,
especially if the whole plaintext has some regularities.

So my idea was - for example - to divide each byte of plaintext
in the middle and to create two bytes from these half bytes.
For example, the byte 0xB6 = 10110110 would lead to
0x_6 = ____0110 and 0x_B = ____1011.

Then I would fill up the upper half of the two bytes with
random bits (under the assumption of course that it IS possible
to create random bits, perhaps by using special hardware
events or devices).

This means that the 4 bits of valuable information in one byte
of plaintext are randomized by the upper 4 bits and that now
one block of valuable(!) plaintext together with the 4 random
bits encrypts to many different blocks of ciphertext.

If the sequence of random bits is really random and equally
distributed then there exist 16 different bytes containing the
same information in the lower 4 bits and that would make up
to 16^8 = 2^32 different blocks of ciphertext for one block
of valuable plaintext.

So I thought that the goal of randomizing the plaintext and
thereby reducing the possibility of creating a code book
could be reached by the method which I proposed, and not
that it would even make a plaintext attack easier. Of course
it would increase (doubling up) the number of blocks a hacker
can look at, but I thought that creating different types of
plaintext blocks would be much more important for security.

Actually I'll prefer the methods that you proposed (encrypting
larger blocks of the file in CBC mode), but I'm still a little
bit confused.

Marco Stolpe

- --
PGP Key, ID 0x4F3FE0B5 should be available on a keyserver
Fingerprint:
D0AA F39C 0D9D 4AC8 D742  C0DB 3536 3D29 4F3F E0B5

-----BEGIN PGP SIGNATURE-----
Version: PGPfreeware 5.0i for non-commercial use
Charset: noconv

iQA/AwUBNoEHoDU2PSlPP+C1EQIKHACdEfv5No3lX3vk1U5H90jdKjESrcoAnihA
BIJtd+Bwk+XQ3jM02wWpGdPv
=Kf3R
-----END PGP SIGNATURE-----
```

```
Subject: Re: Enhancement of EBC mode?
Date: Thu, 24 Dec 1998 06:31:54 GMT
From: ritter@io.com (Terry Ritter)
Message-ID: &LT;3681df9a.18319798@news.io.com>
References: &LT;368109DA.E70A98BD@usa.net>
Newsgroups: sci.crypt
Lines: 35


On Wed, 23 Dec 1998 16:18:50 +0100, in &LT;368109DA.E70A98BD@usa.net>, in
sci.crypt Marco Stolpe &LT;marco.stolpe@usa.net> wrote:

>[...]
>Then I would fill up the upper half of the two bytes with
>random bits (under the assumption of course that it IS possible
>to create random bits, perhaps by using special hardware
>events or devices).
>
>This means that the 4 bits of valuable information in one byte
>of plaintext are randomized by the upper 4 bits and that now
>one block of valuable(!) plaintext together with the 4 random
>bits encrypts to many different blocks of ciphertext.

This technique is disclosed in one of the Feistel patents; oddly, we
do not find it in the crypto texts.

I have been describing it on sci.crypt for the past few years as the
"homophonic" block cipher construction.  It is "homophonic" in that
each different random value selects a different ciphertext for exactly
the same data field.  The "random" field also can be used to validate
the block against possible changes in transit, even if blocks are
delivered out of sequence.

The technique is obviously most efficient when we have large blocks;
Feistel mentions it in the context of 128-bit blocks (Lucifer), but
even that would be expensive.  We need enough "random" bits per block
to make a cryptographic difference.

---
Terry Ritter   ritter@io.com   http://www.io.com/~ritter/
Crypto Glossary   http://www.io.com/~ritter/GLOSSARY.HTM
```

```
Subject: Re: Enhancement of EBC mode?
Date: 23 Dec 1998 23:15:54 GMT
From: ka@socrates.hr.lucent.com.no_spam (Kenneth Almquist)
Message-ID: &LT;75rtja$1s7@nntpa.cb.lucent.com>
References: &LT;367F54AB.173BBA95@usa.net>
Newsgroups: sci.crypt
Lines: 75

> I'm planning to write an application where encryption of files
> is needed but random access to the content of these files is
> necessary.
>
> !- From chapter nine of the book "Applied Cryptography" by
> Bruce Schneier (1996) I concluded that the only thing I can
> do is to use a block cipher with ECB mode.

If you only want to read the data in a random order, you can use
CBC mode.  (To decrypt a block, you have to read the preceding
block as well, which should not be a major problem.)  So I assume
you need to write random locations as well as read them.

You seem to be assuming that the only alternative to CBC mode is
ECB mode.  To understand the other alternatives, it may be helpful
to review the rationale for CBC mode.  The basic weakness of ECB
mode is that a pair of identical plaintext blocks will produce
identical ciphertext blocks.  To avoid this, we need to come up
with an encryption function which encrypts each block differently.
Specifically, we want to come up with a block encryption function

        C = E'(P, K, BN, IV, Prev)
where
        C is the ciphertext block produced by the function.
        P is the plaintext block to encrypt.
        K is the encryption key.
        BN is the block number of the block to encrypt.
        IV is a random value.
        Prev is a vector consisting of all plaintext blocks which precede P.

The first two arguments are the arguments standardly passed to a block
encryption algorithm.  The next argument is BN.  It is used to cause
the encryption function to be different for each block in the file.
The IV argument is needed to prevent block BN of two different files
from being encrypted the same way.  From a security point of view, it
does not matter whether E' uses its last argument.

The latter arguments to E' act like additional key data.  In fact, the
most conceptually simple way to define E' is to define

        E'(P, K, BN, IV, Prev) = E(P, K || BN || IV)

where || is the concatenation operator.  The defintion of E' used with
CBC mode is

        E'(P, K, BN, IV, Prev) = E(P xor H(BN, IV, Prev), K)

H is a hash function.  It is defined as follows:

        function H(BN, IV, Prev) is
            result := IV
            for i := 1 to BN - 1 do
                result := result xor Prev[i]
                result := E(result, K)
            return result

This hash function produces uniformly distributed random numbers if
the block encryption function E is any good.  But the real cleverness
in this choice of hash function is the fact that it is equal to the
result of encrypting the previous block.  Thus there is no need to
actually compute this hash function.

Once you understand the theory behind CBC mode, coming up with an
alternative which allows random access is straightforward.  For example,
you can stick with the hash function approach, but use a hash function
which doesn't depend upon the value of Prev.  One such hash function
is E(BN, IV), giving us:

        E'(P, K, BN, IV, Prev) = E(P xor E(BN, IV), K)

The coresponding decryption function is

        D'(C, K, BN, IV, Prev) = D(C, K) xor E(BN, IV)

Kenneth Almquist
```

```
Subject: Re: Enhancement of EBC mode?
Date: Thu, 24 Dec 1998 06:32:24 GMT
From: ritter@io.com (Terry Ritter)
Message-ID: &LT;3681dfde.18387695@news.io.com>
References: &LT;75rtja$1s7@nntpa.cb.lucent.com>
Newsgroups: sci.crypt
Lines: 40


On 23 Dec 1998 23:15:54 GMT, in &LT;75rtja$1s7@nntpa.cb.lucent.com>, in
sci.crypt ka@socrates.hr.lucent.com.no_spam (Kenneth Almquist) wrote:

>> I'm planning to write an application where encryption of files
>> is needed but random access to the content of these files is
>> necessary.

>[...]
>The basic weakness of ECB
>mode is that a pair of identical plaintext blocks will produce
>identical ciphertext blocks.  To avoid this, we need to come up
>with an encryption function which encrypts each block differently.
>Specificly, we want to come up with a block encryption function
>
>       C = E'(P, K, BN, IV, Prev)
>where
>       C is the ciphertext block produced by the function.
>       P is the plaintext block to encrypt.
>       K is the encryption key.
>       BN is the block number of the block to encrypt.
>       IV is a random value.
>       Prev is a vector consisting of all plaintext blocks which precede P.

Let me point out that it can be a serious mistake to build a system
which uses a file position in encryption.  For example, if the file is
any form of database, it could not then be re-organized, nor could new
blocks be "inserted" in the middle.  So while this solution may be
fine for reading static files or ciphering absolute storage, that same
approach may be inappropriate in a dynamic database.

One alternative is to store a "block number" value along with each
block so that it travels with the block.  But it would be easier to
just save an IV for each block.

---
Terry Ritter   ritter@io.com   http://www.io.com/~ritter/
Crypto Glossary   http://www.io.com/~ritter/GLOSSARY.HTM
```

```
Subject: Re: Enhancement of EBC mode?
Date: Thu, 24 Dec 1998 17:08:44 GMT
From: jsavard@tenMAPSONeerf.edmonton.ab.ca (John Savard)
Message-ID: &LT;368272c2.1212553@news.prosurfr.com>
References: &LT;3681dfde.18387695@news.io.com>
Newsgroups: sci.crypt
Lines: 51

ritter@io.com (Terry Ritter) wrote, in part:

>Let me point out that it can be a serious mistake to build a system
>which uses a file position in encryption.  For example, if the file is
>any form of database, it could not then be re-organized, nor could new
>blocks be "inserted" in the middle.  So while this solution may be
>fine for reading static files or ciphering absolute storage, that same
>approach may be inappropriate in a dynamic database.

>One alternative is to store a "block number" value along with each
>block so that it travels with the block.  But it would be easier to
>just save an IV for each block.

For the application discussed, altering the data to obscure patterns
by XORing each block with its absolute address need not create a
problem;

data is decrypted whenever it is read, and it is encrypted whenever it
is written.

If an application *not authorized to read the data* were, nontheless,
permitted to re-organize the data, then there would be a problem. This
situation, though, does not occur in many applications: in some, for
example, without the decryption key, not only does one have no idea
where record boundaries are, one also does not know which blocks
belong to a file.

Just as in networks there is a difference between end-to-end
encryption and link-to-link encryption, whole-disk and whole-file
encryption can do certain things, and record encryption can do others.

So one could use CBC mode within a record, and one could use the XOR
of an address relative to the start of the file for file encryption,
and one could use the XOR of a track and sector address for disk
encryption without a problem.

Someone with the disk encryption password only could still copy the
file to another disk without destroying it;

someone with the disk encryption password and the file encryption
password could move records within the file;

and someone with all three keys could modify and read data within a
record.

So it depends on the level where encryption is used what measures are
available to mask against codebook attacks without preventing random
access.

John Savard
http://www.freenet.edmonton.ab.ca/~jsavard/index.html
```

```
Subject: Re: Enhancement of EBC mode?
Date: Thu, 24 Dec 1998 19:28:31 GMT
From: ritter@io.com (Terry Ritter)
Message-ID: &LT;368295a6.9181110@news.io.com>
References: &LT;368272c2.1212553@news.prosurfr.com>
Newsgroups: sci.crypt
Lines: 90


On Thu, 24 Dec 1998 17:08:44 GMT, in
&LT;368272c2.1212553@news.prosurfr.com>, in sci.crypt
jsavard@tenMAPSONeerf.edmonton.ab.ca (John Savard) wrote:

>ritter@io.com (Terry Ritter) wrote, in part:
>
>>Let me point out that it can be a serious mistake to build a system
>>which uses a file position in encryption.  For example, if the file is
>>any form of database, it could not then be re-organized, nor could new
>>blocks be "inserted" in the middle.  So while this solution may be
>>fine for reading static files or ciphering absolute storage, that same
>>approach may be inappropriate in a dynamic database.
>
>>One alternative is to store a "block number" value along with each
>>block so that it travels with the block.  But it would be easier to
>>just save an IV for each block.
>
>For the application discussed, altering the data to obscure patterns
>by XORing each block with its absolute address need not create a
>problem;
>
>data is decrypted whenever it is read, and it is encrypted whenever it
>is written.
>
>If an application *not authorized to read the data* were, nontheless,
>permitted to re-organize the data, then there would be a problem. This
>situation, though, does not occur in many applications: in some, for
>example, without the decryption key, not only does one have no idea
>where record boundaries are, one also does not know which blocks
>belong to a file.
>
>Just as in networks there is a difference between end-to-end
>encryption and link-to-link encryption, whole-disk and whole-file
>encryption can do certain things, and record encryption can do others.
>
>So one could use CBC mode within a record, and one could use the XOR
>of an address relative to the start of the file for file encryption,
>and one could use the XOR of a track and sector address for disk
>encryption without a problem.
>
>Someone with the disk encryption password only could still copy the
>file to another disk without destroying it;
>
>someone with the disk encryption password and the file encryption
>password could move records within the file;
>
>and someone with all three keys could modify and read data within a
>record.
>
>So it depends on the level where encryption is used what measures are
>available to mask against codebook attacks without preventing random
>access.

Yes, an encryption sensitivity to physical storage location can be
hidden by deciphering on every read, then enciphering on every write,
but this can have both significant costs and unexpected risks.

Consider, for example, the case of disk encryption, where the
plaintext is randomized by a track / sector hash.  Inevitably there
will come a time when we wish to upgrade storage with a new drive.
Now, upgrading can be tedious even when we do just a straight copy
from drive to drive.  But when the encryption depends upon the track /
sector position, we have to read each "block," decipher it, decide
where to put it, then re-encipher the block on the new drive.  This is
a significant overhead at a bad time, and incidentally implies the
transient exposure of the data as plaintext.

Worse, the inability to simply copy sector contents to new store is a
mortgage on the future and a risk of failure to intervene, since
automatic upgrade utilities will only copy sectors and change file
pointers.  So if there comes a day when the encrypted data is
forgotten and the storage upgraded (which can be as simple as being
away on a trip), the data will simply become usable.  This is not a
risk of losing a record or two; this is the risk of losing an entire
database, with everything inside, and all the effort it took to
construct.

But these costs need not be paid and the risks need not be taken if we
don't first design a system which depends upon physical storage values
which may change in the future.  And there are similar overhead and
transient exposure problems with respect to the reorganization of
database storage within files, when encryption depends upon file
position.

---
Terry Ritter   ritter@io.com   http://www.io.com/~ritter/
Crypto Glossary   http://www.io.com/~ritter/GLOSSARY.HTM
```

```
Subject: Re: Enhancement of EBC mode?
Date: Thu, 24 Dec 1998 16:53:29 GMT
From: jsavard@tenMAPSONeerf.edmonton.ab.ca (John Savard)
Message-ID: &LT;36827058.593902@news.prosurfr.com>
References: &LT;367F54AB.173BBA95@usa.net>
Newsgroups: sci.crypt
Lines: 24

Marco Stolpe &LT;marco.stolpe@usa.net> wrote, in part:

>I'm planning to write an application where encryption of files
>is needed but random access to the content of these files is
>necessary.

>- From chapter nine of the book "Applied Cryptography" by
>Bruce Schneier (1996) I concluded that the only thing I can
>do is to use a block cipher with ECB mode.

As others have noted, that isn't quite true.

As has been noted, if you need random access to the files, but in
units of blocks or records, where a block or record is larger than the
blocksize of the cipher you're using, then you can always use CBC mode
within each block or record.

To conceal patterns in the plaintext, where you actually are
restricted to ECB mode, you can always XOR each plaintext block with
its address (perhaps also trivially encrypted). This avoids having to
expand the plaintext at all.

John Savard
http://www.freenet.edmonton.ab.ca/~jsavard/index.html
```

Terry Ritter, his current address, and his top page.

Last updated: 1999-02-20

Combiner-Type Algorithms

A *Ciphers By Ritter* Page

Comments on codebook and combiner algorithms.

"Codebook" algorithms are block ciphers. "Combiner" algorithms are stream ciphers.

Contents

- 1998-11-02 John Savard: "for 'Combiner style' algorithms to be classed as the other major category of algorithm indicates this field has been rather a busier one behind closed doors...."
- 1998-11-03 Douglas A. Gwyn: "Of course it has!"
- 1998-11-03 John Savard: "I was thinking of Terry Ritter's usage...."
- 1998-11-03 Terry Ritter: "That looks like the same English usage to me: Multiple values are combined into a single result."

```
Subject: Combiner-Type Algorithms (was: Alternative to the Embassy...)
Date: Mon, 02 Nov 1998 18:07:45 GMT
From: jsavard@tenMAPSONeerf.edmonton.ab.ca (John Savard)
Message-ID: &LT;363df3e2.5879047@news.prosurfr.com>
References: &LT;71dtci$rtd$1@server.cntfl.com>
   &LT;3639ef90.860249@news.prosurfr.com>
Newsgroups: sci.crypt,talk.politics.crypto,comp.security.pgp.discuss
Lines: 19

"R H Braddam" &LT;rbraddam@aic-fl.com> wrote, in part:

>There are three
>32-bit four-stage pipelined RISC processors running at 100MHZ. Two of those
>are in the Programmable Cryptographic Processor which has a combined
>throughput of >1200 MIPS. One is optimized for Codebook type algorithms, the
>other for Combiner type algorithms.

Paging Terry Ritter?

Except for my recent "Large-Key Brainstorm", and Terry Ritter's work
with Dynamic Substitution and Latin Squares, I've heard of block
ciphers - which could be called "Codebook style" algorithms, and
stream ciphers. But for "Combiner style" algorithms to be classed as
the other major category of algorithm indicates this field has been
rather a busier one behind closed doors...

John Savard
http://members.xoom.com/quadibloc/index.html
```

```
Subject: Re: Combiner-Type Algorithms (was: Alternative to the Embassy...)
Date: Tue, 03 Nov 1998 00:14:40 GMT
From: "Douglas A. Gwyn" &LT;DAGwyn@null.net>
Message-ID: &LT;363E4A8E.37399E0B@null.net>
References: &LT;363df3e2.5879047@news.prosurfr.com>
Newsgroups: sci.crypt,talk.politics.crypto,comp.security.pgp.discuss
Lines: 9

John Savard wrote:
> ... But for "Combiner style" algorithms to be classed as
> the other major category of algorithm indicates this field has been
> rather a busier one behind closed doors...

Of course it has!
"Combiner" is the component that takes bits from the state
of a stream cipher system and "combines" the bits, usually
these days in some nonlinear way, to compute a key bit.
```

```
Subject: Re: Combiner-Type Algorithms (was: Alternative to the Embassy...)
Date: Tue, 03 Nov 1998 16:30:40 GMT
From: jsavard@tenMAPSONeerf.edmonton.ab.ca (John Savard)
Message-ID: &LT;363f2efa.273062@news.prosurfr.com>
References: &LT;363E4A8E.37399E0B@null.net>
Newsgroups: sci.crypt
Lines: 24

"Douglas A. Gwyn" &LT;DAGwyn@null.net> wrote, in part:
>John Savard wrote:

>> ... But for "Combiner style" algorithms to be classed as
>> the other major category of algorithm indicates this field has been
>> rather a busier one behind closed doors...

>Of course it has!

I know cryptography in general is busier behind closed doors than
otherwise; that's only reasonable, since the major users are those who
that serves. I meant that the specific field of that type of algorithm
was busier in relative terms - compared to the concentration on block
ciphers in public work.

>"Combiner" is the component that takes bits from the state
>of a stream cipher system and "combines" the bits, usually
>these days in some nonlinear way, to compute a key bit.

Ah. I was thinking of Terry Ritter's usage, where the combiner is what
applies the key bits to the plaintext to produce ciphertext.

John Savard
http://members.xoom.com/quadibloc/index.html
```

```
Subject: Re: Combiner-Type Algorithms (was: Alternative to the Embassy...)
Date: Tue, 03 Nov 1998 18:50:40 GMT
From: ritter@io.com (Terry Ritter)
Message-ID: &LT;363f5057.10172037@news.io.com>
References: &LT;363f2efa.273062@news.prosurfr.com>
Newsgroups: sci.crypt
Lines: 53


On Tue, 03 Nov 1998 16:30:40 GMT, in
&LT;363f2efa.273062@news.prosurfr.com>, in sci.crypt
jsavard@tenMAPSONeerf.edmonton.ab.ca (John Savard) wrote:

>"Douglas A. Gwyn" &LT;DAGwyn@null.net> wrote, in part:
>[...]
>>"Combiner" is the component that takes bits from the state
>>of a stream cipher system and "combines" the bits, usually
>>these days in some nonlinear way, to compute a key bit.
>
>Ah. I was thinking of Terry Ritter's usage, where the combiner is what
>applies the key bits to the plaintext to produce ciphertext.

That looks like the same English usage to me:  Multiple values are
combined into a single result.

* Combiners can be nonlinear and irreversible, and so can be used to
combine RNG sequences to produce a stronger sequence.

* Combiners are commonly exclusive-OR, which is linear and reversible;
this supports data with running key in a stream cipher, but of course
the combining has no strength at all under known-plaintext.

* Another is Dynamic Substitution, which is nonlinear yet reversible,
and so can be used for stream cipher data / running-key combining, and
yet also have strength under known-plaintext.


I know "combiner" from the patent literature, and the earliest
references largely describe linear and reversible combining (e.g.,
exclusive-OR):

* We see a vacuum tube exclusive-OR in Smith 2,496,317 (1950 Feb 7)
titled "Combining Circuit."

* We see another vacuum tube exclusive-OR in Kohler (1951 Sep 11)
titled "Combining Circuits."

* We also see a *mechanical* exclusive-OR (for teletype encryption) in
Arko 3,159,712 (1961 Aug 30) titled "Mechanical Signal Combiner."

* In body text I happened to come upon Blasbalg 3,715,508 (1973 Feb 6)
titled "Switching Circuits Employing Orthogonal and Quasi-Orthogonal
Pseudo-Random Code Sequences."  In col 3, line 28, we see: "Combiner 7
may be of any well known type such as a multiplier for sampled analog
data or a modulo-two adder for digital data."

---
Terry Ritter   ritter@io.com   http://www.io.com/~ritter/
Crypto Glossary   http://www.io.com/~ritter/GLOSSARY.HTM
```

Terry Ritter, his *current address*, and his *top page*.

Last updated: 1999-01-19

The Homophonic Block Cipher Construction

A *Ciphers By Ritter* Page

A discussion of noise in plaintext, homophonic ciphers, large blocks and other constructions. Also "What is a block cipher?" and FFT mixing and Mixing Cipher descriptions.

(Unfortunately, the first message in this thread did not arrive or was not saved.)

Contents

- 1998-09-30 Harvey Rook: "...the point of vulnerability in almost all well designed crypto-systems is the password, and not the stream."
- 1998-09-30 W T Shaw: "A minor increase in length, a few percent at most with an appropriate algorithm, can include information that allows much stronger encryption than maintaining plain/ciphertext block size at unity."
- 1998-10-02 Bryan G. Olson; CMSC (G): "Rivest and Sherman...suggest applying an error correction code to the plaintext, then randomly changing as many bits as the code can correct."
- 1998-10-02 Mok-Kong Shen: "Could you say whether this is different in principle (spirit) from the classical technique of homophones?"
- 1998-10-10 Scott Nelson: "Adding noise to each and every block isn't done mainly because it isn't necessary. In some cases it can actually be harmful...."
- 1998-10-22 Bo Dömstedt: "One example would be the breaking of the German Enigma by the small Polish bureau [1], exploiting the non-ranomness if initializing vectors."
- 1998-10-02 bryan.olson@uptronics.com: "...there's little evidence that it's needed."
- 1998-10-03 Terry Ritter: "...something like this is achieved in the homophonic block ciphers I have been describing here for the past several years (the basic idea is Feistel, mid-70's):..."
- 1998-10-05 dianelos@tecapro.com: "Still, the method you describe is not sufficient as a defense against known or chosen plaintext attacks...."
- 1998-10-05 Terry Ritter: "there *is* no known-plaintext or defined-plaintext attack in the usual sense...."
- 1998-10-06 dianelos@tecapro.com: "Ciphertext only attacks will be possible though...."
- 1998-10-06 John Savard: "You can, however, think of the padding step as an additional encryption step...."
- 1998-10-06 Terry Ritter: "Surely we can imagine taking half of every 64-bit DES block and filling that with noise. Now, for any 32-bit data (in the other half of the block), there are 2**32 different ciphertexts which represent the exact same data under the exact same DES key!"
- 1998-10-07 Jerry Leichter: "You haven't gained anything at all!"
- 1998-10-08 Terry Ritter: "First, homophonic keying is a randomization which is effective against codebook attack...." "Next, a homophonic field also can supply 'authentication'...."
- 1998-10-08 Jerry Leichter: "This has been known for many years (*not* a criticism)."
- 1998-10-08 Terry Ritter: "Fine, let's see some references (*not* a criticism). My reference is one of the Feistel patents, mid 70's, which an examiner was kind enough to point out as prior art to one of my claims."
- 1998-10-09 Jerry Leichter: "I used such a thing in a stream cipher system I developed in the early '80's, and never thought I was inventing anything."
- 1998-10-10 Terry Ritter: This last smirking comment would be a lot more amusing if: a) Horst Feistel had not thought the homophonic construction was new; b) IBM patent lawyers had not thought it was new; c) The PTO examiner had not thought it was new; and c) A patent had not been granted on it."
- 1998-10-12 Jerry Leichter: "As always with Mr. Ritter, attempts at reasoned discussion eventually turn into selective quotations and insults."
- 1998-10-11 Bryan G. Olson; CMSC (G): "I'm not sure what specific result you're addressing, but I'll once again recommended the paper "Randomized Encryption Techniques" by Ronald Rivest and Alan Sherman from Crypto 82. Of course it post-dates Feistel, but pre-dates discussions on sci.crypt."
- 1998-10-12 Mok-Kong Shen: "This is clearly a homophonic substitution."
- 1998-10-13 Terry Ritter: "But a paper is not a text, and the fact that a 1982 reference has not made it into the crypto texts should tell us just how well "known" that reference and this construction really are."
- 1998-10-14 Bryan G. Olson; CMSC (G): "...if counter mode is secure, then homophonic block ciphers are not needed."
- 1998-10-14 Terry Ritter: "False. The homophonic construction has more utility than counter mode:...."
- 1998-10-14 Bryan G. Olson; CMSC (G): "Did you miss the direction of the implication? Randomized block ciphers are at least as secure as counter mode."
- 1998-10-15 Terry Ritter: "Is the homophonic block cipher construction presented in Applied Cryptography? Is it in the Handbook of Applied Cryptography? If not, the technique is hardly likely to be known by the ordinary reader of those texts."
- 1998-10-15 John Savard: "AC doesn't mention, however, the idea of adding a few extra bits to every block, either for variation (like an IV) or for authentication."
- 1998-10-17 Terry Ritter: "I believe I pointed out previously that the error-correcting code with errors example was indeed a homophonic coding. But it is not a *secret* coding; it is not a homophonic cipher. But we can accomplish the same end *without* using an error-correcting code. Which means we have an advance, folks."
- 1998-10-18 W T Shaw: "...the MAC's that I generate are additive; I seamlessly add the results from sequential blocks together."
- 1998-10-17 Bryan G. Olson; CMSC (G): "...homophones do not provide a means to authenticate. Redundancy, not randomness, offers authentication."
- 1998-10-17 Terry Ritter: "...each of these different fields may be *used* in different ways. Yet they have *exactly* *the* *same* *construction*." "If you want to call them 'randomness' and 'redundancy,' feel free. But don't imagine that you have the only two terms that can be properly used."
- 1998-10-18 Bryan G. Olson; CMSC (G): "The issue is what goes into the field, not what we call it. If it's random - additional entropy - nondeterministic, that means it induces homophones. If it's redundant - deterministic, then and only then is it useful for authentication."
- 1998-10-18 W T Shaw: "You need not add the redundancy to get authentication, merely treat the string as if it had had it added."
- 1998-10-14 W T Shaw: "...the more output possibilities in encryptive mode, the stronger it is in MAC mode."
- 1998-10-12 Mok-Kong Shen: "If the the random 64-bit block is encrypted with another key, wouldn't that be a good scheme?"
- 1998-10-12 Mok-Kong Shen: "I agree with you on the desirability of larger block sizes but wonder why it has apparently not been stressed in the literature as it deserves."
- 1998-10-12 Terry Ritter: "My guess is that the avoidance of large blocks is first related to the use of Feistel constructions for block ciphers."
- 1998-10-13 Paul Crowley: "I'm in the process of designing a very fast large block cipher based around a balanced Feistel network...."
- 1998-10-13 Mok-Kong Shen: "I don't understand your 'it pretty much rules out a bijective F-function'."
- 1998-10-13 Terry Ritter: "...it looks like the main nonlinearity in the system is the single 8 to 32 bit 'T' table, and that makes me nervous."
- 1998-10-16 Mok-Kong Shen: "In Schneier's book there is only a reference to FFT-Hash. Evidently you mean something different. Could you give the relevant references?"
- 1998-10-17 Terry Ritter: "Most crypto scientists see a block cipher as an emulated huge Simple Substitution. So if a 'stream cipher' had that characteristic, it would not *be* a stream cipher...." "We can also see that stream ciphers only diffuse changes to later cipherings."
- 1998-10-17 W T Shaw: "I accept that as a great definition."
- 1998-10-19 Mok-Kong Shen: "Taking the block as the fundamental unit CBC is a stream cipher. So we agree."
- 1998-10-19 W T Shaw: "I wrote some block ciphers that have nothing to do with bits some time ago; it is based on trits."
- 1998-10-19 Mok-Kong Shen: "I mean both character and bit, depending on one's preference, can serve as the fundamental unit in discourse."
- 1998-10-19 Terry Ritter: "In figure 1 we have a typical Mixing cipher in schematic form...."
- 1998-10-20 John Savard: "...the number of web sites is growing at a rate the search engines are finding it hard to keep up with. As a result, they're indexing a smaller proportion of existing Web sites, and a larger number of out-of-date links."

- 1998-10-20 Mok-Kong Shen: "...the grouping of fundamential units into blocks is 'syntatic'. The opearation done on a block is 'semantic' and is entirely dependent on the cipher in question."
- 1998-10-08 Mok-Kong Shen: "I see that I haven't yet properly understood your idea."
- 1998-10-08 dianelos@tecapro.com: "Here is what I want: A method that uses ciphers as a primitive and stops all known plaintext or ciphertext only attacks." "Terry Ritter proposes to include a random field in the text block. This does have advantages but it does not stop ciphertext only attacks...."
- 1998-10-12 Mok-Kong Shen: "As to the topic of chosen plaintext attack I think it is the reference point that is essential."
- 1998-10-12 dianelos@tecapro.com: "O.K. Let us suppose we have two encryption boxes, each of which includes a true random generator."
- 1998-10-13 Mok-Kong Shen: "You are right. A PRNG can hardly compete with a real TRNG. (There are some difficulties of obtaining and employing really good TRNGs though, in my humble opinion.)"

```
Subject: Re: Thought question:  why is encrypted output routinely the same size as
the input?
Date: Wed, 30 Sep 1998 11:03:14 -0700
From: "Harvey Rook" &LT;RedRook@ZippyTheYahoo.com>
Message-ID: &LT;6utrjf$jk1@news.dns.microsoft.com>
References: &LT;36125A50.47C3@sundialservices.com>
Newsgroups: sci.crypt
Lines: 28

Check out Rivest's Chaffing and Winnowing scheme.

On a more practical note, the point of vulnerability in almost all well
designed crypto-systems is the password, and not the stream. Right now we
(probably, but not provably) have ciphers that can only be attacked by brute
force, yet these ciphers don't increase the size of the stream.

So, if the real weakness in a crypto-system is the password, then increasing
the stream size by adding random noise doesn't increase your security, it
only wastes band width.

Harvey Rook
Spam Guard. Remove the "Zippy The" to send email.
RedRook@ZippyTheYahoo.com



Sundial Services wrote in message &LT;36125A50.47C3@sundialservices.com>...
>One thing that puzzles me about all of the encryption designs that I
>know of is that none of them seem to use the notion of deliberately
>injecting noise into the stream...  stuff that is really not part of the
>plaintext at all but that is enciphered along with it.
>
>It seems to me that, with the obvious and acceptable side-effect of
>making the ciphertext larger than the plaintext, this would add greatly
>to the troubles of a cryptanalyst.  Why isn't this done?
```

```
Subject: Re: Thought question:  why is encrypted output routinely the same size as
the input?
Date: Wed, 30 Sep 1998 19:34:00 -0600
From: jgfunj@EnqvbSerrGrknf.pbz (W T Shaw)
Message-ID: &LT;jgfunj-3009981934000001@dialup178.itexas.net>
References: &LT;6utpvj$32c$1@quine.mathcs.duq.edu>
   &LT;36125A50.47C3@sundialservices.com>
Newsgroups: sci.crypt
Lines: 32

In article &LT;6utpvj$32c$1@quine.mathcs.duq.edu>, juola@mathcs.duq.edu
(Patrick Juola) wrote:

> In article &LT;36125A50.47C3@sundialservices.com>,
> Sundial Services  &LT;info@sundialservices.com> wrote:
> >One thing that puzzles me about all of the encryption designs that I
> >know of is that none of them seem to use the notion of deliberately
> >injecting noise into the stream...  stuff that is really not part of the
> >plaintext at all but that is enciphered along with it.
>
> Where do you get the noise?  With a well-designed cypher, all the
> "noise" you need is part of the key.
>
A minor increase in length, a few percent at most with an appropriate
algorithm, can include information that allows much stronger encryption
than maintaining plain/ciphertext block size at unity.  This has nothing
to do with noise such as nulls meant to confuse the analysit, or with
inefficient algorithms where block size is doubled or more.

Consideration is given to use of such an algorithm when used to produce a
MAC, where a minimal amount of information lost also insures that the
strength of the algorithm is maintained in that mode.  Nothing is really
lost, as the few missing characters are simply used, integrated into the
key structure, a part of it predetermined by the operator.  This may be as
hard to convey as following only oral instructions on how to tie your
shoes.
--
----
Show me a politician who does not lie through his teeth,
and.....I'll show you one who can't find his dentures.
----
Decrypt with ROT13 to get correct email address.
```

Subject: Re: Thought question: why is encrypted output routinely the same size as the input?
Date: 2 Oct 1998 02:11:13 GMT
From: olson@umbc.edu (Bryan G. Olson; CMSC (G))
Message-ID: <6v1co1$ip5$1@news.umbc.edu>
References: <36130001.33B5045A@null.net>
<36125A50.47C3@sundialservices.com>
Newsgroups: sci.crypt
Lines: 26

Douglas A. Gwyn (DAGwyn@null.net) wrote:
: Sundial Services wrote:
: > One thing that puzzles me about all of the encryption designs that I
: > know of is that none of them seem to use the notion of deliberately
: > injecting noise into the stream...

: This *is* done in certain schemes, but it isn't so simple as you
: suggest.
: In general, either the "noise" needs to be cryptographically generated,
: using a key so that the intended recipient can undo it,
: or the recipient needs to apply some sort of general (unkeyed) noise
: filter.

I suppose it's not exactly simple, but a technique proposed
by Rivest and Sherman falls into the second category, and it
isn't all that complex. They suggest applying an error
correction code to the plaintext, then randomly changing as
many bits as the code can correct.

: Your apparent intent can be met better by reducing redundancy before
: encrypting, which also makes better use of the channel bandwidth.

Not if the intent is to harden the system against known or
chosen plaintext attacks.

--Bryan

Subject: Re: Thought question: why is encrypted output routinely the same size as the input?
Date: Fri, 02 Oct 1998 16:19:08 +0100
From: Mok-Kong Shen <mok-kong.shen@stud.uni-muenchen.de>
Message-ID: <3614EEEC.482F00A@stud.uni-muenchen.de>
References: <6v1co1$ip5$1@news.umbc.edu>
Newsgroups: sci.crypt
Lines: 12

Bryan G. Olson; CMSC (G) wrote:

> I suppose it's not exactly simple, but a technique proposed
> by Rivest and Sherman falls into the second category, and it
> isn't all that complex. They suggest applying an error
> correction code to the plaintext, then randomly changing as
> many bits as the code can correct.

Could you say whether this is different in principle (spirit)
from the classical technique of homophones?

M. K. Shen

Subject: Re: Thought question: why is encrypted output routinely the same size as the input?
Date: Sat, 10 Oct 1998 18:47:27 GMT
From: scott@helsbreth.org (Scott Nelson)
Message-ID: <361fa045.3520234@news.inreach.com>
References: <36125A50.47C3@sundialservices.com>
Newsgroups: sci.crypt
Lines: 31

On Wed, 30 Sep, Sundial Services <info@sundialservices.com> wrote:

>One thing that puzzles me about all of the encryption designs that I
>know of is that none of them seem to use the notion of deliberately
>injecting noise into the stream... stuff that is really not part of the
>plaintext at all but that is enciphered along with it.
>
>It seems to me that, with the obvious and acceptable side-effect of
>making the ciphertext larger than the plaintext, this would add greatly
>to the troubles of a cryptanalyst. Why isn't this done?
>
It is done. In particular, Cipher Block Chaining
starts with 1 block of IV, in essence increasing
the ciphertext by one block, and randomizing all
the plaintext.

Adding noise to each and every block isn't done mainly
because it isn't necessary. In some cases it
can actually be harmful, consider:

The encryption is DES with 56 bits of random noise
in the first 7 bytes, and 1 byte of real data.

Mallet collects 256 unique plain texts.
He can now forge any message he wants to.

Shameless plug for random web site:
http://www.helsbreth.org/random
Scott Nelson <scott@helsbreth.org>

Subject: Re: Thought question: why is encrypted output routinely the same size as the input?
Date: Thu, 22 Oct 1998 13:39:21 GMT
From: bo.doemstedt@mbox200.swipnet.se (Bo Dömstedt)
Message-ID: <362f2d50.515707890@nntpserver.swip.net>
References: <36125A50.47C3@sundialservices.com>
Newsgroups: sci.crypt
Lines: 54

Sundial Services <info@sundialservices.com> wrote:
>One thing that puzzles me about all of the encryption designs that I
>know of is that none of them seem to use the notion of deliberately
>injecting noise into the stream... stuff that is really not part of the
>plaintext at all but that is enciphered along with it.
>
>It seems to me that, with the obvious and acceptable side-effect of
>making the ciphertext larger than the plaintext, this would add greatly
>to the troubles of a cryptanalyst. Why isn't this done?

Your question refer to "old stuff" that are now very well understood.
One example would be the breaking of the German Enigma by
the small Polish bureau [1], exploiting the non-ranomness
if initializing vectors. A systematic study of ciphers and
randomness can be found in [2].

>Why isn't this done?
Well, if you don't, could your system be broken using
depth reading ??

Bo Dömstedt
Cryptographer
Protego Information AB
Malmoe,Sweden

SG100 true random noise generator
http://www.protego.se/sg100_en.htm

--- Run your own OTP! 750 MB/24hrs!
--- Generate initialization vectors
--- Generate session keys
--- Generate encryption keys
--- Provide randomness for public key generation.

[1]
Rejewski, Marian
"How Polish mathematicians deciphered the Enigma"
Annals of the History of Computing Vol 3 No 3 July 1981, pp 213-229
translated by American Federation of Information Processing from
"Jak matematycy polscy rozszyfrowali Enigme"
Annals of the Polihs Mathematical Society, Series II
Wiadomosci Matematyczne, Volume 23, 1980, 1-28

[2]
Rivest, Ronald L. and Sherman, Alan T.
"Randomized Encryption Techniques"
Chaum, David; Rivest, Ronald L. and Sherman, Alan T. (editors)
pages 145-164
"Advances in Cryptology Proceedings of Crypto 82"
Plenum Press New York 1982
ISBN 0-306-41366-3

Subject: Re: Thought question: why is encrypted output routinely the same size as the input?
Date: Fri, 02 Oct 1998 01:56:44 GMT
From: bryan.olson@uptronics.com
Message-ID: <6v1bss$ir3$1@nnrp1.dejanews.com>
References: <36125A50.47C3@sundialservices.com>
Newsgroups: sci.crypt
Lines: 26

  info@sundialservices.com wrote:
> One thing that puzzles me about all of the encryption designs that I
> know of is that none of them seem to use the notion of deliberately
> injecting noise into the stream... stuff that is really not part of the
> plaintext at all but that is enciphered along with it.

As I recall (i.e. I haven't looked it up), the proceedings of
Crypto 82 contained three papers that proposed adding
randomness to symmetric ciphers. The most comprehensive was
Rivest and Sherman's "Randomized Encryption Techniques".

Adding randomness seems especially powerful for resisting

chosen plaintext attacks.

> It seems to me that, with the obvious and acceptable side-effect of
> making the ciphertext larger than the plaintext, this would add greatly
> to the troubles of a cryptanalyst. Why isn't this done?

Primarily because there's little evidence that it's needed.
Modern Ciphers used with a simple unique IV seem to resist
every attack that randomized ciphers resist.

--Bryan

----------== Posted via Deja News, The Discussion Network ==----------
http://www.dejanews.com/ Search, Read, Discuss, or Start Your Own

---

Subject: Re: Thought question: why is encrypted output routinely the same size as the input?
Date: Sat, 03 Oct 1998 19:39:15 GMT
From: ritter@io.com (Terry Ritter)
Message-ID: <36167d50.2983728@news.io.com>
References: <6v5mcq$cj1$1@nnrp1.dejanews.com>
  <3613D2D3.9F597176@stud.uni-muenchen.de>
Newsgroups: sci.crypt
Lines: 83

On Sat, 03 Oct 1998 17:20:28 GMT, in
<6v5mcq$cj1$1@nnrp1.dejanews.com>, in sci.crypt dianelos@tecapro.com
wrote:

>[...]
> The original scheme of including noise in the encryption process
> may have important practical value. I believe that the following
> is true:
>
> If a) one true random bit is injected for each bit encrypted and
> b) the encryption function fulfills some "weak" mathematical
> condition, then the resulting encryption method cannot be broken
> by *any* chosen plaintext, known plaintext of ciphertext only
> attack, in other words offers practically perfect security. The
> only attacks that work would be exhaustive key search and, maybe,
> related key, an attack difficult to curry out in practice.
>
> The "weak" mathematical condition mentioned above is, roughly,
> that there exists no efficient method to compute R based on E(R) and
> E(T xor R) when T is known or chosen and R is a random number.
> This seems to be a much weaker condition to prove than what cipher
> designers would have to prove for an almost perfect cipher: that
> even if a large number of T, E(T) pairs are known there is no way
> to compute the secret key. Observe that for no cipher has this
> last condition been proved - cipher designers can only claim that
> a cipher has resisted much effort to find such a method, not that
> such a method does not exist.
>
> I think this "weak" mathematical condition is so weak that
> probably any modern cipher fulfills it. If somebody would prove
> this then we could design practical, virtually impregnable
> security systems, albeit doubling the size of the ciphertext.

Allow me to point out that something like this is achieved in the
homophonic block ciphers I have been describing here for the past
several years (the basic idea is Feistel, mid-70's):

1. Reserve an authentication / keying field in the data block (this
of course reduces the amount of data which will fit in the block).

2. Place a random value in that field.

3. Encipher.

Note that each possible value in the a/k field produces a completely
different ciphertext block. But if we decipher *any* of those blocks,
we find that the data field is the same. We can use this is lots of
ways:

a. Simply as a homophonic block cipher, using truly random a/k. (We
can assure that the Opponents will not collect a useful codebook from
ECB operation, so we can use ECB, and we don't have to transport or
create an IV to do it.)

b. As a block self-authentication, by using an arbitrary a/k value.
(Every time we decipher a block, we can compare the a/k we get to the
one used originally, or just to the a/k from each block, so if The
Opponent changes a block, the a/k won't check out.)

c. As a dynamically-keyed cipher, with zero keying setup.
(Essentially change the key on a block-by-block basis.)

Note that (b) & (c) can be accomplished simultaneously, and will
effectively produce (a) as well.

I claim this keying is like any block keying in that enough is enough.
If we have 64 bits of a/k (or 80 or whatever), we don't need more. Of

course, a 64-bit a/k field is 50% of a 128-bit block, but just 12.5% of a 64 byte block. This is one of the advantages of a large block construction.

I think the essence of this sort of field is the "block-ness" of the transformation, the guarantee that changing the a/k field will change the transformation from any tractable subset of bits to the output. If the transformation does not change for, say, a byte, that byte has not been dynamically keyed.

Terry Ritter ritter@io.com http://www.io.com/~ritter/
Crypto Glossary 1998-08-27: http://www.io.com/~ritter/GLOSSARY.HTM

Subject: Re: Thought question: why is encrypted output routinely the same size as the input?
Date: Mon, 05 Oct 1998 06:48:47 GMT
From: dianelos@tecapro.com
Message-ID: <6v9q4e$n10$1@nnrp1.dejanews.com>
References: <36167d50.2983728@news.io.com>
Newsgroups: sci.crypt
Lines: 75

In article <36167d50.2983728@news.io.com>,
ritter@io.com (Terry Ritter) wrote:
>
> On Sat, 03 Oct 1998 17:20:28 GMT, in
> <6v5mcq$cj1$1@nnrp1.dejanews.com>, in sci.crypt dianelos@tecapro.com
> wrote:
>
> >[...]
> > The original scheme of including noise in the encryption process
> > may have important practical value. I believe that the following
> > is true:
> >
> > If a) one true random bit is injected for each bit encrypted and
> > b) the encryption function fulfills some "weak" mathematical
> > condition, then the resulting encryption method cannot be broken
> > by *any* chosen plaintext, known plaintext of ciphertext only
> > attack, in other words offers practically perfect security. The
> > only attacks that work would be exhaustive key search and, maybe,
> > related key, an attack difficult to curry out in practice.
> >
> > The "weak" mathematical condition mentioned above is, roughly,
> > that there exists no efficient method to compute R based on E(R) and
> > E(T xor R) when T is known or chosen and R is a random number.
> > This seems to be a much weaker condition to prove than what cipher
> > designers would have to prove for an almost perfect cipher: that
> > even if a large number of T, E(T) pairs are known there is no way
> > to compute the secret key. Observe that for no cipher has this
> > last condition been proved - cipher designers can only claim that
> > a cipher has resisted much effort to find such a method, not that
> > such a method does not exist.
> >
> > I think this "weak" mathematical condition is so weak that
> > probably any modern cipher fulfills it. If somebody would prove
> > this then we could design practical, virtually impregnable
> > security systems, albeit doubling the size of the ciphertext.
>
> Allow me to point out that something like this is achieved in the
> homophonic block ciphers I have been describing here for the past
> several years (the basic idea is Feistel, mid-70's):
>
> 1. Reserve an authentication / keying field in the data block (this
> of course reduces the amount of data which will fit in the block).
>
> 2. Place a random value in that field.
>
> 3. Encipher.
>
> Note that each possible value in the a/k field produces a completely
> different ciphertext block. But if we decipher *any* of those blocks,
> we find that the data field is the same. We can use this is lots of
> ways: [...]

Yes. Still, the method you describe is not sufficient as a defense against known or chosen plaintext attacks because the rest of the data block can be known or chosen.

Even if the a/k field is half the block-size and is xor-ed to the other half before encryption, you still leak information. For example if an attacker chooses data filled with zeros, he knows that the block that will be enciphered has two identical halves.

Compare this to the method I propose:

```
E( k1,  E( k2, R ) )
E( k3,  E( k4, T xor R ) )
```

It is inefficient both in time and space, but can you see any way to mount an attack on this apart from exhaustive key search?

--
http://www.tecapro.com
email: dianelos@tecapro.com

----------== Posted via Deja News, The Discussion Network ==----------
http://www.dejanews.com/ Search, Read, Discuss, or Start Your Own

---

Subject: Re: Thought question: why is encrypted output routinely the same size as the input?
Date: Mon, 05 Oct 1998 18:09:41 GMT
From: ritter@io.com (Terry Ritter)
Message-ID: <36190b22.6139167@news.io.com>
References: <6v9q4e$n10$1@nnrp1.dejanews.com>
Newsgroups: sci.crypt
Lines: 138

On Mon, 05 Oct 1998 06:48:47 GMT, in
<6v9q4e$n10$1@nnrp1.dejanews.com>, in sci.crypt dianelos@tecapro.com
wrote:

>In article <36167d50.2983728@news.io.com>,
> ritter@io.com (Terry Ritter) wrote:
>>
>> On Sat, 03 Oct 1998 17:20:28 GMT, in
>> <6v5mcq$cj1$1@nnrp1.dejanews.com>, in sci.crypt dianelos@tecapro.com
>> wrote:
>>
>> >[...]
>> > The original scheme of including noise in the encryption process
>> > may have important practical value. I believe that the following
>> > is true:
>> >
>> > If a) one true random bit is injected for each bit encrypted and
>> > b) the encryption function fulfills some "weak" mathematical
>> > condition, then the resulting encryption method cannot be broken
>> > by *any* chosen plaintext, known plaintext of ciphertext only
>> > attack, in other words offers practically perfect security. The
>> > only attacks that work would be exhaustive key search and, maybe,
>> > related key, an attack difficult to curry out in practice.
>> >
>> > The "weak" mathematical condition mentioned above is, roughly,
>> > that there exists no efficient method to compute R based on E(R) and
>> > E(T xor R) when T is known or chosen and R is a random number.
>> > This seems to be a much weaker condition to prove than what cipher
>> > designers would have to prove for an almost perfect cipher: that
>> > even if a large number of T, E(T) pairs are known there is no way
>> > to compute the secret key. Observe that for no cipher has this
>> > last condition been proved - cipher designers can only claim that
>> > a cipher has resisted much effort to find such a method, not that
>> > such a method does not exist.
>> >
>> > I think this "weak" mathematical condition is so weak that
>> > probably any modern cipher fulfills it. If somebody would prove
>> > this then we could design practical, virtually impregnable
>> > security systems, albeit doubling the size of the ciphertext.
>>
>> Allow me to point out that something like this is achieved in the
>> homophonic block ciphers I have been describing here for the past
>> several years (the basic idea is Feistel, mid-70's):
>>
>> 1. Reserve an authentication / keying field in the data block (this
>> of course reduces the amount of data which will fit in the block).
>>
>> 2. Place a random value in that field.
>>
>> 3. Encipher.
>>
>> Note that each possible value in the a/k field produces a completely
>> different ciphertext block. But if we decipher *any* of those blocks,
>> we find that the data field is the same. We can use this is lots of
>> ways: [...]
>
> Yes. Still, the method you describe is not sufficient as a defense
> against known or chosen plaintext attacks because the rest of the
> data block can be known or chosen.

Obviously I have been unable to communicate what the a/k stuff is.

Maybe:

```
   C[i] = E( k, (P[i] &LT;&LT; 64) + a/k[i]) )
```

where E is a 64 byte block cipher
C[i] is 64 byte ciphertext
P[i] is 56 byte plaintext
a/k[i] is 8 byte (64 bit) authentication / keying value

We can assume that every possible value for a/k will produce a different ciphertext, and that every ciphertext bit will have an equal opportunity to change.

This means there will be a multiplicity of ciphertext representations

for the exact same data: this is a homophonic block cipher. And that means:

1) there *is* no known-plaintext or defined-plaintext attack in the usual sense; the opponent can only see or set the subset of the block which is the data field; the a/k field is internal;

2) if a/k is essentially random-like, we can use Electronic CodeBook (ECB) mode, instead of a chaining mode (this means that blocks can be enciphered and deciphered independently); (here we discard the a/k field upon deciphering);

3) if we produce a sequence of a/k values, we are essentially providing dynamic keying for each block, and if we can reproduce the a/k sequence in deciphering (just like any stream cipher), we can *also* check the validity of each block and the sequence of blocks.

```
>    Even if the a/k field is half the block-size and is xor-ed to the
>    other half before encryption,
```

No, a/k functions differently. It is related to your idea by concept, not implementation.

```
>you still leak information. For
>    example if an attacker chooses data filled with zeros, he knows
>    that the block that will be enciphered has two identical halves.
```

No. This is not a/k.

```
>    Compare this to the method I propose:
>
>       E( k1,  E( k2, R ) )
>       E( k3,  E( k4, T xor R ) )
>
>     It is inefficient both in time and space, but can you see any
>     way to mount an attack on this apart from exhaustive key search?
```

At the top we encipher R twice, under two different keys. At the bottom we encipher T xor R twice, under yet two other keys.

But if enciphering is effective, why do we do it twice? And if it is not effective, why is twice enough?

And if we are going to attack E(), we will have to know what E() is: We have passed the point of notational utility.

Summary

If we had a wide block cipher we could effectively add block keying to the data stream. And that would make size restrictions on the primary key essentially useless.

Terry Ritter ritter@io.com http://www.io.com/~ritter/
Crypto Glossary 1998-08-27: http://www.io.com/~ritter/GLOSSARY.HTM

Subject: Re: Thought question: why is encrypted output routinely the same size as the input?
Date: Tue, 06 Oct 1998 04:23:20 GMT
From: dianelos@tecapro.com
Message-ID: <6vc5vn$6fs$1@nnrp1.dejanews.com>
References: <36190b22.6139167@news.io.com>
Newsgroups: sci.crypt
Lines: 92

In article <36190b22.6139167@news.io.com>,
ritter@io.com (Terry Ritter) wrote:

```
>[...]
>Dianelos wrote:
> >    Yes. Still, the method you describe is not sufficient as a defense
> >    against known or chosen plaintext attacks because the rest of the
> >    data block can be known or chosen.
>
> Obviously I have been unable to communicate what the a/k stuff is.
>
> Maybe:
>
>    C[i] = E( k, (P[i] &LT;&LT; 64) + a/k[i]) )
>
> where E is a 64 byte block cipher
>       C[i] is 64 byte ciphertext
>       P[i] is 56 byte plaintext
>       a/k[i] is 8 byte (64 bit) authentication / keying value
>
> We can assume that every possible value for a/k will produce a
> different ciphertext, and that every ciphertext bit will have an equal
> opportunity to change.
>
```

> This means there will be a multiplicity of ciphertext representations
> for the exact same data: this is a homophonic block cipher. And that
> means:
>
> 1) there *is* no known-plaintext or defined-plaintext attack in the
> usual sense; the opponent can only see or set the subset of the block
> which is the data field; the a/k field is internal;

O.K. I suppose the usual sense of known-plaintext is that the whole plaintext is known to the attacker. Ciphertext only attacks will be possible though, because these only require partial knowledge of the plaintext.

>[...]
> > Compare this to the method I propose:
> >
> > E(k1, E(k2, R))
> > E(k3, E(k4, T xor R))
> >
> > It is inefficient both in time and space, but can you see any
> > way to mount an attack on this apart from exhaustive key search?
>
> At the top we encipher R twice, under two different keys. At the
> bottom we encipher T xor R twice, under yet two other keys.
>
> But if enciphering is effective, why do we do it twice? And if it is
> not effective, why is twice enough?

Because if single encryption is used then the noise block R can be canceled out. Suppose we use:

```
E( k1, R ) = C1
E( k2, T xor R ) = C2
```

where C1, C2 are the two ciphertext blocks.

Now the attacker computes:

```
R = D( k1, C1 )
T xor R = D( k2, C2 )
=> T xor D( k1, C1 ) = D( k2, C2 )
=> T = D( k1, C1 ) xor D( k2, C2 )
```

The noise factor has now disappeared. If the plaintexts are known, the attacker can accumulate a large number of relations where the only unknowns are k1 and k2.

For similar reasons, the following method will not work either:

```
E( k1,  E( k2, R ) ) = C1
E( k3,  T xor R ) = C2
```

It seems we need two double encryptions.

> And if we are going to attack E(), we will have to know what E() is:
> We have passed the point of notational utility.

Suppose it is E=DES. My point is that any good cipher will probably work. At the cost of quadrupling time and doubling space it appears we can get a cipher where the only possible attack is exhaustive key search. If E=DES this is an impossible 2^224 work.

--
http://www.tecapro.com
email: dianelos@tecapro.com

----------== Posted via Deja News, The Discussion Network ==----------
http://www.dejanews.com/ Search, Read, Discuss, or Start Your Own

---

Subject: Re: Thought question: why is encrypted output routinely the same size as the input?
Date: Tue, 06 Oct 1998 16:11:08 GMT
From: jsavard@tenMAPSONeerf.edmonton.ab.ca (John Savard)
Message-ID: <361a3f1a.5668671@news.prosurfr.com>
References: <6vc5vn$6fs$1@nnrp1.dejanews.com>
Newsgroups: sci.crypt
Lines: 43

dianelos@tecapro.com wrote, in part:

> O.K. I suppose the usual sense of known-plaintext is that the
> whole plaintext is known to the attacker. Ciphertext only attacks
> will be possible though, because these only require partial
> knowledge of the plaintext.

You can, however, think of the padding step as an additional encryption step - so you know the plaintext before that step, even if you can't do a true known-plaintext attack on the following step by itself.

So a known-plaintext attack remains possible in a technical or terminological sense, while operationally one *may* be able to prevent

a known-plaintext attack on a portion of the encipherment process that is vulnerable to it.

However, let us say one is facing single-DES as a cipher. A ciphertext-only attack is not known, but a known-plaintext one is: brute force.

Now then, if I use CBC mode, I fail to eliminate a known-plaintext attack, because if one knows the plaintext, one knows the input to DES, the XOR of the plaintext and the previous ciphertext block.

However, I could use a different mode that does eliminate a direct known-plaintext attack on DES, and yet is still insecure:

send, enciphered, a random 64-bit block, then XOR that block to every block in the message before encryption.

I don't know, now, the value of any plaintext block. But I can still brute-force search: decipher a *pair* of blocks for every key, and when the difference between them, deciphered, is the same as that between the two plaintext blocks, I probably have the right key.

This is a trivial example, but it illustrates the principle that a method that _apparently_ eliminates a true known-plaintext attack may not actually give security to a cipher, even if that cipher is not vulnerable in the case of unknown plaintext.

John Savard
http://members.xoom.com/quadibloc/index.html

Subject: Re: Thought question: why is encrypted output routinely the same size as the input?
Date: Tue, 06 Oct 1998 19:15:48 GMT
From: ritter@io.com (Terry Ritter)
Message-ID: <361a6bfd.13557766@news.io.com>
References: <361a3f1a.5668671@news.prosurfr.com>
Newsgroups: sci.crypt
Lines: 37

On Tue, 06 Oct 1998 16:11:08 GMT, in
<361a3f1a.5668671@news.prosurfr.com>, in sci.crypt
jsavard@tenMAPSONeerf.edmonton.ab.ca (John Savard) wrote:

>[...]
>This is a trivial example, but it illustrates the principle that a
>method that _apparently_ eliminates a true known-plaintext attack may
>not actually give security to a cipher, even if that cipher is not
>vulnerable in the case of unknown plaintext.

One could presumably make a similar argument that not all keying is worthwhile, because some forms of keying are ineffective.

Surely we can imagine taking half of every 64-bit DES block and filling that with noise. Now, for any 32-bit data (in the other half of the block), there are 2**32 different ciphertexts which represent the exact same data under the exact same DES key! Known-plaintext is now 32 bits of data with 64 bits of ciphertext.

Then the question is: "How many different DES keys can produce the same plaintext-to-ciphertext transformation (given some homophonic keying value)?" Well, *some* plaintext must produce a given ciphertext under *every* key, so the chance that it has our particular data is just the size of that field, 1 in 2**32. So with a 56-bit DES key, we expect that fully 2**24 different DES keys will have the exact same plaintext-to-ciphertext transformation.

Placing keying in the data block is the homophonic construction, and if we have a bigger block, we can have more keying and less overhead.

Terry Ritter ritter@io.com http://www.io.com/~ritter/
Crypto Glossary 1998-08-27: http://www.io.com/~ritter/GLOSSARY.HTM

Subject: Re: Thought question: why is encrypted output routinely the same size as the input?
Date: Wed, 07 Oct 1998 11:09:55 -0400
From: Jerry Leichter <leichter@smarts.com>
Message-ID: <361B8443.32DF@smarts.com>
References: <361a6bfd.13557766@news.io.com>
Newsgroups: sci.crypt
Lines: 51

| Surely we can imagine taking half of every 64-bit DES block and
| filling that with noise. Now, for any 32-bit data (in the other half
| of the block), there are 2**32 different ciphertexts which represent
| the exact same data under the exact same DES key! Known-plaintext is
| now 32 bits of data with 64 bits of ciphertext.
|

| Then the question is: "How many different DES keys can produce the
| same plaintext-to-ciphertext transformation (given some homophonic
| keying value)?" Well, *some* plaintext must produce a given
| ciphertext under *every* key, so the chance that it has our particular
| data is just the size of that field, 1 in 2**32. So with a 56-bit DES
| key, we expect that fully 2**24 different DES keys will have the exact
| same plaintext-to-ciphertext transformation.

Fine. Now how many of those keys also give a reasonable decryption of the next block? Let's continue with your worst-case analysis: 1 in 2^32 keys give the right plaintext for one block. Since everything here is independent, only 1 in 2^64 keys gives the right plaintext for any two distinct blocks. Since there are only 2^56 keys, it's highly likely that the only key that actually gives the right plaintext for two blocks is the one being sought. Note that this is true even assuming the block key value is chosen completely at random for each block.

So, it would seem as if you've doubled the known plaintext required. In fact, however, that's an illusion: What've you've doubled is the *amount of ciphertext corresponding to known plaintext*. However, each block of ciphertext now only encodes half the plaintext it did before. Originally, you needed one 64-bit block of known plaintext. Now you need to 32-bit blocks of known plaintext. You haven't gained anything at all!

This argument is independent of the number of bytes you reserve for keying material, and the size of the block. All you're doing is spreading the known material out through a number of blocks. There's a bit more work, but not all that much. (At the very worst, if you use k blocks, you make the opponent do k times as many trial decryptions - actually, less, since he can often stop early. However, the user of the system also has to do k times as many encryptions and decryptions, so you haven't increased the work factor for the cryptanalyst at all.)

The underlying problem is fundamental: The cleartext and the random filler material are trivially separable once the external encipherment has been removed. So they don't really add anything in defending against a brute-force attack: In a brute-force attack, the attacker has his hands on the results of removing the external encipherment, along with a lot of random junk that he has to discard. You haven't changed the work needed in the brute force step - the external encipherment keyspace is exactly the size it originally was - and you haven't made it significantly harder for the attacker to discard the junk.

-- Jerry

Subject: Re: Thought question: why is encrypted output routinely the same size as the input?
Date: Thu, 08 Oct 1998 07:09:31 GMT
From: ritter@io.com (Terry Ritter)
Message-ID: <361c6524.53243250@news.io.com>
References: <361B8443.32DF@smarts.com>
Newsgroups: sci.crypt
Lines: 138

On Wed, 07 Oct 1998 11:09:55 -0400, in <361B8443.32DF@smarts.com>, in sci.crypt Jerry Leichter <leichter@smarts.com> wrote:

>| Surely we can imagine taking half of every 64-bit DES block and
>| filling that with noise. Now, for any 32-bit data (in the other half
>| of the block), there are 2**32 different ciphertexts which represent
>| the exact same data under the exact same DES key! Known-plaintext is
>| now 32 bits of data with 64 bits of ciphertext.
>|
>| Then the question is: "How many different DES keys can produce the
>| same plaintext-to-ciphertext transformation (given some homophonic
>| keying value)?" Well, *some* plaintext must produce a given
>| ciphertext under *every* key, so the chance that it has our particular
>| data is just the size of that field, 1 in 2**32. So with a 56-bit DES
>| key, we expect that fully 2**24 different DES keys will have the exact
>| same plaintext-to-ciphertext transformation.
>
>Fine. Now how many of those keys also give a reasonable decryption of
>the next block?

Any time we have more data than key, *any* cipher system is "theoretically solvable." If you were expecting this construction to repeal Shannon, I am sorry to say it does not.

But if you just want more keyspace, that is simple enough to do: Just add another cipher with its own large keyspace and multi-cipher.

>Let's continue with your worst-case analysis: 1 in
>2^32 keys give the right plaintext for one block. Since everything here
>is independent, only 1 in 2^64 keys gives the right plaintext for any
>two distinct blocks. Since there are only 2^56 keys, it's highly likely
>that the only key that actually gives the right plaintext for two blocks
>is the one being sought. Note that this is true even assuming the block
>key value is chosen completely at random for each block.
>
>So, it would seem as if you've doubled the known plaintext required. In

>fact, however, that's an illusion: What've you've doubled is the
>*amount of ciphertext corresponding to known plaintext*. However, each
>block of ciphertext now only encodes half the plaintext it did before.
>Originally, you needed one 64-bit block of known plaintext. Now you
>need to 32-bit blocks of known plaintext. You haven't gained anything
>at all!

Since this was not the intended gain, we should not be surprised to
not achieve it. If adding keyspace to DES was this easy, we would
have heard about it 20 years ago.

The real "illusion" here is the assumption that homophonic keying is
in all cases equivalent to the cipher main key. But if one is going
to use every new construction exactly like the old one, it is going to
be very difficult to deliver any useful new features. There is more
to cryptography than just doing DES again, with a bigger block and a
larger key.

The homophonic field can oppose codebook attacks. Is this "keying"?
Well, we normally think that each key value will generate a different
ciphertext, and that is exactly what the homophonic field does. So
the homophonic field *is* "keying," even if it is a little different
than usual.

>This argument is independent of the number of bytes you reserve for
>keying material, and the size of the block. All you're doing is
>spreading the known material out through a number of blocks. There's a
>bit more work, but not all that much. (At the very worst, if you use k
>blocks, you make the opponent do k times as many trial decryptions -
>actually, less, since he can often stop early. However, the user of the
>system also has to do k times as many encryptions and decryptions, so
>you haven't increased the work factor for the cryptanalyst at all.)
>
>The underlying problem is fundamental: The cleartext and the random
>filler material are trivially separable once the external encipherment
>has been removed. So they don't really add anything in defending
>against a brute-force attack: In a brute-force attack, the attacker has
>his hands on the results of removing the external encipherment, along
>with a lot of random junk that he has to discard. You haven't changed
>the work needed in the brute force step - the external encipherment
>keyspace is exactly the size it originally was - and you haven't made it
>significantly harder for the attacker to discard the junk.

We should note that you somehow failed to take note of the next step,
as it was in my original:

>>Placing keying in the data block is the homophonic construction,
>>and if we have a bigger block, we can have more keying and less
>>overhead.

The DES example was explanatory; the advantages accrue mainly with use
in block ciphers with large blocks (which minimize homophonic
overhead) and large main keys (which eliminate the need to expand the
main keyspace).

I see two real advantages in the homophonic construction:

1. First, homophonic keying is a randomization which is effective
against codebook attack: When a plaintext block is repeated with a
different homophonic value it will have a different ciphertext. (The
homophonic keying value can be really random in this use.)

And codebook attacks are important enough that most systems defend
against them with something beyond the cipher per se, typically "CBC
mode." The homophonic keying field thus provides a way to avoid CBC
chaining and the sequentiality that implies. (Only huge blocks -- say
64 bytes or larger -- would have enough uniqueness from text to avoid
codebook attack without randomizing the plaintext or the transform.)

Sequentiality means that throughput generally cannot be improved with
multiple ciphering hardware, and that lost or delayed packets will
delay ciphering even if later packets are already received and ready.
And the longer the round trip, the worse the consequences. These are
problems we can avoid with homophonic keying.

2. Next, a homophonic field also can supply "authentication" --
provided we know what value the field should have upon deciphering.
Possibly the authentication value could be related to the message key,
so that any block received for that message would give the same
authentication value. Perhaps another part of the field could hold
the message block number.

Indeed, with transport level ciphering, the authentication value might
even be used as the network error-detection code (for ciphered packets
only), thus avoiding perhaps *two* error-detection passes across the
data.

There are advantages to the homophonic construction (and large blocks
and variable-size blocks) to the extent that the cipher is allowed to
do what it can do and is not forced into the cramped and primitive
DES-style box by which all ciphers are currently judged. The new
advantages *can* be significant -- in particular applications.

Terry Ritter ritter@io.com http://www.io.com/~ritter/
Crypto Glossary 1998-08-27: http://www.io.com/~ritter/GLOSSARY.HTM

Subject: Re: Thought question: why is encrypted output routinely the same size as the input?
Date: Thu, 08 Oct 1998 10:54:41 -0400
From: Jerry Leichter <leichter@smarts.com>
Message-ID: <361CD231.2B3B@smarts.com>
References: <361c6524.53243250@news.io.com>
Newsgroups: sci.crypt
Lines: 76

Terry Ritter wrote:
[A number of interesting things.]

Basically, I don't disagree with any of it. The *context* in which it was posted made it appear as if Mr. Ritter was proposing to use homophonic keying as a way to increase the effective cost of a brute-force attack (which *is* a goal of Rivest's all-or-nothing construction). As I pointed out, that won't work.

| I see two real advantages in the homophonic construction:
|
| 1. First, homophonic keying is a randomization which is effective
| against codebook attack: When a plaintext block is repeated with a
| different homophonic value it will have a different ciphertext. (The
| homophonic keying value can be really random in this use.)
|
| And codebook attacks are important enough that most systems defend
| against them with something beyond the cipher per se, typically "CBC
| mode." The homophonic keying field thus provides a way to avoid CBC
| chaining and the sequentiality that implies. (Only huge blocks -- say
| 64 bytes or larger -- would have enough uniqueness from text to avoid
| codebook attack without randomizing the plaintext or the transform.)

This has been known for many years (*not* a criticism). A common thing to throw at the inexperienced, after they've convinced themselves that RSA, say, is "mathematically unbreakable independent of message content", is: "OK, so you want to send me a one-word message - either YES or NO. How do you do it?" The "obvious" answer - pad with 0's the block size and encrypt - of course fails miserably against a trivial codebook attack. So, where did the "mathematical security" go?

Note that for the particular proposed method to work, your encryption algorithm need to have rather strong properties - "semantic security" and other similar notions. (At the least, all the bits must be "equally strong". *Some* of the bits of the input are known to have this property in RSA - the low bit, something like the top log n bits. Combinatorial ciphers *seem* on their face to treat all bits the same, and security against differential crypto guarantees you that certain kinds of information about the input must be hidden to some known degree, but beyond that ... it's an interesting question.)

| 2. Next, a homophonic field also can supply "authentication" --
| provided we know what value the field should have upon deciphering.
| Possibly the authentication value could be related to the message key,
| so that any block received for that message would give the same
| authentication value. Perhaps another part of the field could hold
| the message block number.

However, uses 1 and 2 contradict each other to some degree. If the homophonic field is constant within a session, an attacker can build a dictionary for that session. Not nearly as good as a global dictionary, but it may well reveal useful information. Then again, if you're using a session key, even *without* the HF, the dictionary is only good for one session - so what have you gained? So the HF has to change fairly regularly. You could, of course, change the session key for the same reason - though admittedly for many algorithms this is much more expensive.

I'm not sure I understand how the homophonic field would be used as an authentication value. If it's a pre-agreed constant, then you get no help against a dictionary attack. If it's computed by some method, and that method is attackable, then if an attacker manages to break one block, he may be able to attack the HF generation algorithm and thus forge further blocks. Could you talk more about how you'd compute an HF field for authentication?

| Indeed, with transport level ciphering, the authentication value might
| even be used as the network error-detection code (for ciphered packets
| only), thus avoiding perhaps *two* error-detection passes across the
| data.

On the other hand, this *helps* brute-force (or brute-force-like) attacks: It provides a quick test of whether you've gotten the right key, without knowing anything at all about the plaintext.

-- Jerry

Subject: Re: Thought question: why is encrypted output routinely the same size as the

input?
Date: Thu, 08 Oct 1998 22:10:49 GMT
From: ritter@io.com (Terry Ritter)
Message-ID: <361d385a.9708199@news.io.com>
References: <361CD231.2B3B@smarts.com>
Newsgroups: sci.crypt
Lines: 190

On Thu, 08 Oct 1998 10:54:41 -0400, in <361CD231.2B3B@smarts.com>, in
sci.crypt Jerry Leichter <leichter@smarts.com> wrote:

>Terry Ritter wrote:
>[A number of interesting things.]
>
>Basically, I don't disagree with any of it. The *context* in which it
>was posted made it appear as if Mr. Ritter was proposing to use
>homophonic keying as a way to increase the effective cost of a brute-
>force attack (which *is* a goal of Rivest's all-or-nothing construc-
>tion). As I pointed out, that won't work.

Upon rereading, I agree that it was misleading.

Part of my point was that block ciphers already function similarly to
all-or-nothing, with respect to mixing within their particular block.
And if we can inject noise into the plaintext block, known-plaintext
effectively goes away.

I think the idea of injecting noise into the plaintext was part of
this thread and that prompted my entry into the discussion.

>| I see two real advantages in the homophonic construction:
>|
>| 1. First, homophonic keying is a randomization which is effective
>| against codebook attack: When a plaintext block is repeated with a
>| different homophonic value it will have a different ciphertext. (The
>| homophonic keying value can be really random in this use.)
>|
>| And codebook attacks are important enough that most systems defend
>| against them with something beyond the cipher per se, typically "CBC
>| mode." The homophonic keying field thus provides a way to avoid CBC
>| chaining and the sequentiality that implies. (Only huge blocks -- say
>| 64 bytes or larger -- would have enough uniqueness from text to avoid
>| codebook attack without randomizing the plaintext or the transform.)
>
>This has been known for many years (*not* a criticism).

Fine, let's see some references (*not* a criticism).

My reference is one of the Feistel patents, mid 70's, which an
examiner was kind enough to point out as prior art to one of my
claims. As far a I know I introduced the idea on sci.crypt several
years ago in discussions about my large block designs, You
participated in some of those discussions, and some of those are
archived on my pages. I am not aware of a text reference.

The main problem with the homophonic construction is that it eats bits
in the block. So unless we have a sizable block, we can't have a
sizable homophonic field, and even then it means per-block overhead,
so we need a pretty big block to make all this efficient.

>A common thing
>to throw at the inexperienced, after they've convinced themselves that
>RSA, say, is "mathematically unbreakable independent of message
>content", is: "OK, so you want to send me a one-word message - either
>YES or NO. How do you do it?" The "obvious" answer - pad with 0's the
>block size and encrypt - of course fails miserably against a trivial
>codebook attack. So, where did the "mathematical security" go?

This is why DES is almost always used in some form of plaintext
randomization such as CBC. The reason for using CBC is also not
covered well in texts.

>Note that for the particular proposed method to work, your encryption
>algorithm need to have rather strong properties - "semantic security"
>and other similar notions. (At the least, all the bits must be "equally
>strong". *Some* of the bits of the input are known to have this
>property in RSA - the low bit, something like the top log n bits.
>Combinatorial ciphers *seem* on their face to treat all bits the same,
>and security against differential crypto guarantees you that certain
>kinds of information about the input must be hidden to some known
>degree, but beyond that ... it's an interesting question.)

Well, this seems to be what we expect from a block cipher anyway.

Presumably, if we can get a statistical bias which is independent of
key, we can look into the plaintext. But the real problem is a
statistical bias to the key, and if there is any, we can start to
develop that key. So this is bad, but I am unaware of any reasoning
which would prove statistical key independence for any block cipher.
The possibility of doing this is one of the hoped-for but as yet
unrealized advantages of the Mixing cipher construction.

Personally, I think "counter mode" is dangerous for just this reason in any Feistel structure, and would always work to avoid the situation where the total plaintext change could be a single increment.

>| 2. Next, a homophonic field also can supply "authentication" --
>| provided we know what value the field should have upon deciphering.
>| Possibly the authentication value could be related to the message key,
>| so that any block received for that message would give the same
>| authentication value. Perhaps another part of the field could hold
>| the message block number.
>
>However, uses 1 and 2 contradict each other to some degree.

Sure.

>If the
>homophonic field is constant within a session, an attacker can build a
>dictionary for that session.

Yes. Don't do that.

>Not nearly as good as a global dictionary,
>but it may well reveal useful information. Then again, if you're using
>a session key, even *without* the HF, the dictionary is only good for
>one session - so what have you gained? So the HF has to change fairly
>regularly.

Well, *part* of it does, by which I imply that part need not.

I guess the problem here is that one field, with a single homophonic property, can be partitioned and used in different ways, and so becomes conceptually different fields. We can use one, the other, or both simultaneously.

The whole concept is like that; the only fields in the data block are what we agree to have. The cipher sees it as just more data. And we just interpret that data differently.

>You could, of course, change the session key for the same
>reason - though admittedly for many algorithms this is much more
>expensive.
>
>I'm not sure I understand how the homophonic field would be used as an
>authentication value. If it's a pre-agreed constant, then you get no
>help against a dictionary attack.

Then just imagine that we have a separate field for authentication. We may also have a field for homophonic keying.

>If it's computed by some method, and
>that method is attackable, then if an attacker manages to break one
>block, he may be able to attack the HF generation algorithm and thus
>forge further blocks. Could you talk more about how you'd compute an HF
>field for authentication?

I think there are many possibilities with various tradeoffs.

For authentication only, I like a hash of the message key, or part of that key itself. This gives a particular authentication value which is fixed for all blocks in a particular message, and we can check that value on each block after deciphering.

For randomization, I like some sort of really random or at least unknowable value, different for each block. We just throw that away when deciphering.

Both authentication and randomization can be used simultaneously in separate fields.

A different authentication alternative is to use a really random value which is fixed for all blocks, and then just require that each block have the same value -- so we compare block to block, but without inducing sequentiality. Again, the authentication value is fixed for all blocks.

Another alternative is to use a LFSR or LCG to produce values for subsequent blocks, and this means that the same field can be used for both authentication and randomization, but this returns us to hated sequentiality.

It may be that in many cases, with a large enough block, only authentication is needed.

>| Indeed, with transport level ciphering, the authentication value might
>| even be used as the network error-detection code (for ciphered packets
>| only), thus avoiding perhaps *two* error-detection passes across the
>| data.
>
>On the other hand, this *helps* brute-force (or brute-force-like)
>attacks: It provides a quick test of whether you've gotten the right
>key, without knowing anything at all about the plaintext.

Well, yeah, but the only time we can really use this stuff is if we

have large blocks (and presumably large keys). We are thus past brute-force threat anyway.

Terry Ritter ritter@io.com http://www.io.com/~ritter/
Crypto Glossary 1998-08-27: http://www.io.com/~ritter/GLOSSARY.HTM

Subject: Re: Thought question: why is encrypted output routinely the same size as the input?
Date: Fri, 09 Oct 1998 15:17:18 -0400
From: Jerry Leichter <leichter@smarts.com>
Message-ID: <361E613E.1862@smarts.com>
References: <361d385a.9708199@news.io.com>
Newsgroups: sci.crypt
Lines: 128

| >[Homophonic keying]
| >This has been known for many years (*not* a criticism).
|
| Fine, let's see some references (*not* a criticism).

We may have different notions of what "This" refers to. However ... the classic paper on this topic is:

> Shafi Goldwasser, Silvio Micali, and Po Tong. Why and how to establish a private code on a public network (extended abstract). In 23rd Annual Symposium on Foundations of Computer Science, pages 134-144, Chicago, Illinois, 3-5 November 1982. IEEE. Citations.

This is among the earliest crypto papers written by two people who later became very active in the crypto community, Shafi Goldwasser and Silvio Micali. I don't know what happened to Po Tong - it's the only paper with his name on it ever to appear at FOCS. It's been years since I read this paper, so I may be mis-remembering it, but I believe one of the thing it discusses is the problem of sending a single bit securely. This is a kind of least-common-denominator problem; if you can do this, you can do many things safely. And it immediately forces you to look at attacks on the system, not just the mathematics. Their proposal to deal with it is "probabilistic encryption", which is homophonic keying under another name: For each plaintext, there must be (very) many possible cryptotexts. The actual way they *implement* this is very different from Mr. Ritter's proposal, however.

The same idea obviously occurs the "salt" used in Unix password encryption, and for that matter in the choice of a random IV in CBC encryption. All of these are related to the indicators used in systems going back at least to World War I. (An indicator was an extra bit of keying material, chosen - nominally at random - for each message, and sent along with the message in the clear. It's whole purpose is to keep the opponent from seeing any two messages encrypted with exactly the same key. As best I can recall, Kahn mentions this in The Code-breakers.) I used such a thing in a stream cipher system I developed in the early '80's, and never thought I was inventing anything.

The idea of using random *padding* has been proposed too many times to track down; and nonces are used in many protocols for authentication in the style Mr. Ritter proposes. However, I will not claim that I've previously seen the *specific* configuration Mr. Ritter discusses, in which the "indicator" is used, not to change the key, but to perturb each block of a block cipher. This may very well be new.

| >I'm not sure I understand how the homophonic field would be used as
| >an authentication value. If it's a pre-agreed constant, then you get
| >no help against a dictionary attack.
|
| Then just imagine that we have a separate field for authentication.
| We may also have a field for homophonic keying.
|
| >If it's computed by some method, and
| >that method is attackable, then if an attacker manages to break one
| >block, he may be able to attack the HF generation algorithm and thus
| >forge further blocks. Could you talk more about how you'd compute an
| >HF field for authentication?
|
| I think there are many possibilities with various tradeoffs....
|
| For authentication only, I like a hash of the message key, or part of
| that key itself. This gives a particular authentication value which
| is fixed for all blocks in a particular message, and we can check that
| value on each block after deciphering....
|
| A different authentication alternative is to use a really random value
| which is fixed for all blocks, and then just require that each block
| have the same value -- so we compare block to block, but without
| inducing sequentiality. Again, the authentication value is fixed for
| all blocks.

Someone who doesn't know the session key can't produce cryptotext that will decrypt to anything reasonable. (If, in such a protocol, it's hard to tell if something is "reasonable" - i.e., there is almost no redundancy in valid messages - then there are many ways to add such redundancy. This is certainly one way, though perhaps not the best.)

On the other hand, someone who *does* have the key can decrypt a message and will then know the authentication value. This strikes me as a very weak approach. (BTW, recall Matt Blaze's attack against the Clipper LEAF algorithm. It used this kind of approach, though it was vulnerable because its authenticator was only 16 bits long.)

| Another alternative is to use a LFSR or LCG to produce values for
| subsequent blocks, and this means that the same field can be used for
| both authentication and randomization, but this returns us to hated
| sequentiality....

If you're going to do that, why not just used a MAC? That has its own internal strength; if it uses a key independent of the session key, it provides protection even if the attacker manages to get the session key. "Sequentiality" is an issue only if you make it one - the MAC can be over as large or as small a part of the data as you like. In practice, for most purposes there's a semantic "message size" that's much longer than one block, and little point to using a MAC over things shorter than one meaningful message. However, if you really want it, a per-block MAC is indeed reasonable if the block size is large enough.

| >| Indeed, with transport level ciphering, the authentication value
| >|might even be used as the network error-detection code (for ciphered
| >|packets only), thus avoiding perhaps *two* error-detection passes
| >|across the data.
| >
| >On the other hand, this *helps* brute-force (or brute-force-like)
| >attacks: It provides a quick test of whether you've gotten the right
| >key, without knowing anything at all about the plaintext.
|
| Well, yeah, but the only time we can really use this stuff is if we
| have large blocks (and presumably large keys). We are thus past
| brute-force threat anyway.

Pure brute-force, yes - independent of block size, any cryptosystem designed today can easily, and should, be designed to make pure brute-force attacks impossible. However, speaking abstractly, there may very well be attacks that limit the possible keyspace by analytic means, but still require a huge (but now possible) search over what's left. All sorts of mathematical results come down to "case analysis", and computers have already been used where there are huge number of cases to try out. Differential crypto has this form, of course - though it doesn't look at a large number of possible keys, but at a large number of key *differentials*. Anyway, if there were such an attack on an algorithm, an internal checksum would help it. (On the other hand, an internal *MAC* with a secret key wouldn't help at all. There's an interesting tradeoff between an internal MAC (of the cleartext) and an external MAC (of the ciphertext): If the opponent learns the MAC key, but not the cipher key, for an internal MAC, he might have a brute-force -like attack; while for an external key, he could create fake ciphertext (though he wouldn't be able to control what it decrypted to).)

-- Jerry

Subject: Re: Thought question: why is encrypted output routinely the same size as the input?
Date: Sat, 10 Oct 1998 08:03:43 GMT
From: ritter@io.com (Terry Ritter)
Message-ID: <361f145f.3782561@news.io.com>
References: <361E613E.1862@smarts.com>
Newsgroups: sci.crypt
Lines: 55

On Fri, 09 Oct 1998 15:17:18 -0400, in <361E613E.1862@smarts.com>, in sci.crypt Jerry Leichter <leichter@smarts.com> wrote:

>| >[Homophonic keying]
>| >This has been known for many years (*not* a criticism).
>|
>| Fine, let's see some references (*not* a criticism).

>[...]
>The same idea obviously occurs the "salt" used in Unix password
>encryption, and for that matter in the choice of a random IV in CBC
>encryption. All of these are related to the indicators used in systems
>going back at least to World War I. (An indicator was an extra bit of
>keying material, chosen - nominally at random - for each message, and
>sent along with the message in the clear. It's whole purpose is to keep
>the opponent from seeing any two messages encrypted with exactly the
>same key. As best I can recall, Kahn mentions this in The Code-
>breakers.) I used such a thing in a stream cipher system I developed in
>the early '80's, and never thought I was inventing anything.

This last smirking comment would be a lot more amusing if:

a) Horst Feistel had not thought the homophonic construction was new;
b) IBM patent lawyers had not thought it was new;
c) The PTO examiner had not thought it was new; and
c) A patent had not been granted on it.

This of course happened in the mid-70's, so if Jerry had been just a few years earlier, he could have intervened, because surely anybody who ever made a stream cipher with a key indicator would know how to

make a homophonic block cipher.

There is a fundamental difference between the homophonic construction and the simple randomization of a salt, or a message key, or the CBC IV: In the homophonic construction, each possible plaintext "letter" has multiple unique ciphertext "codes" which represent that letter. This is the classic meaning of a "homophonic" cipher, and the implication is that each "letter" has its own unique subset of ciphertext codes. But CBC and other randomization techniques are not confined in this way, and in general can take any plaintext to any possible ciphertext. This is a striking difference, and one implication is that a randomized cipher cannot be uniquely deciphered without extra information (such as the preceding-block ciphertext, in CBC mode), but a homophonic cipher *can*. This is block independence, and it can be useful.

If someone else has comments on this, perhaps we could have a more reasonable discussion.

Terry Ritter ritter@io.com http://www.io.com/~ritter/
Crypto Glossary 1998-08-27: http://www.io.com/~ritter/GLOSSARY.HTM

Subject: Re: Thought question: why is encrypted output routinely the same size as the input?
Date: Mon, 12 Oct 1998 11:16:45 -0400
From: Jerry Leichter <leichter@smarts.com>
Message-ID: <36221D5D.1628@smarts.com>
References: <361f145f.3782561@news.io.com>
Newsgroups: sci.crypt
Lines: 47

| >The same idea obviously occurs the "salt" used in Unix password
| >encryption, and for that matter in the choice of a random IV in CBC
| >encryption. All of these are related to the indicators used in
| >systems going back at least to World War I. (An indicator was an
| >extra bit of keying material, chosen - nominally at random - for each
| >message, and sent along with the message in the clear. It's whole
| >purpose is to keep the opponent from seeing any two messages
| >encrypted with exactly the same key. As best I can recall, Kahn
| >mentions this in The Codebreakers.) I used such a thing in a stream
| >cipher system I developed in the early '80's, and never thought I was
| >inventing anything.
|
| This last smirking comment would be a lot more amusing if:
| [Irrelevent stuff]

As always with Mr. Ritter, attempts at reasoned discussion eventually turn into selective quotations and insults.

I never thought what *I* was original because I simply applied ideas taken from public sources, many of which I listed.

The *very next paragraph*, which Mr. Ritter chose not to quote, read:

> The idea of using random *padding* has been proposed too many times to track down; and nonces are used in many protocols for authentication in the style Mr. Ritter proposes. However, I will not claim that I've previously seen the *specific* configuration Mr. Ritter discusses, in which the "indicator" is used, not to change the key, but to perturb each block of a block cipher. This may very well be new.

Mr. Ritter's indicates that it is, indeed, new. Fine; congratulations to Mr. Ritter on developing a clever, original idea. (Absolutely *no* condescention intended, though I'm sure Mr. Ritter will manage to take offense, regardless of what I say here.)

The very *first* paragraph of my message - also not quoted read:

> We may have different notions of what "This" refers to.

Given all the additional discussion in Mr. Ritter's message about CBC, I *still* don't understand exactly what it is *he* thinks our disagreement is about. Since he has a patent, perhaps he can cite the patent number - or, even better, provide the text of the patent or one of his white papers that describes what he has that's new.

I will say nothing further on this subject.

-- Jerry

Subject: Re: Thought question: why is encrypted output routinely the same size as the input?
Date: 11 Oct 1998 23:55:29 GMT
From: olson@umbc.edu (Bryan G. Olson; CMSC (G))
Message-ID: <6vrghhroq1@news.umbc.edu>
References: <361d385a.9708199@news.io.com>
Newsgroups: sci.crypt
Lines: 30

Terry Ritter (ritter@io.com) wrote:

: My reference is one of the Feistel patents, mid 70's, which an
: examiner was kind enough to point out as prior art to one of my
: claims. As far a I know I introduced the idea on sci.crypt several
: years ago in discussions about my large block designs, You
: participated in some of those discussions, and some of those are
: archived on my pages. I am not aware of a text reference.

I'm not sure what specific result you're addressing, but
I'll once again recommended the paper "Randomized Encryption
Techniques" by Ronald Rivest and Alan Sherman from Crypto 82.
Of course it post-dates Feistel, but pre-dates discussions
on sci.crypt.

[...]
: Personally, I think "counter mode" is dangerous for just this reason
: in any Feistel structure, and would always work to avoid the situation
: where the total plaintext change could be a single increment.

I think the same intuition supports Rivest and Sherman's idea
of adding randomness by applying an error correction code, and
then flipping as many bits as the code can correct. Unlike
the simple random field, it doesn't tell the attacker the location
of known and unknown plaintext bits.

--Bryan

Subject: Re: Thought question: why is encrypted output routinely the same size as the
input?
Date: Mon, 12 Oct 1998 14:09:53 +0100
From: Mok-Kong Shen <mok-kong.shen@stud.uni-muenchen.de>
Message-ID: <3621FFA1.5C10765A@stud.uni-muenchen.de>
References: <6vrghhroq1@news.umbc.edu>
Newsgroups: sci.crypt
Lines: 18

Bryan G. Olson; CMSC (G) wrote:
>
> Terry Ritter (ritter@io.com) wrote:
> > ....

> I think the same intuition supports Rivest and Sherman's idea
> of adding randomness by applying an error correction code, and
> then flipping as many bits as the code can correct. Unlike
> the simple random field, it doesn't tell the attacker the location
> of known and unknown plaintext bits.

This is clearly a homophonic substitution. The subset of all code
words that are error-corrected to one code word map to that one code
word, i.e. a many-to-one (homophonic) mapping. (Note that one assumes
that the transmission is error free now, which is certainly given
on the application level.)

M. K. Shen

Subject: Re: Thought question: why is encrypted output routinely the same size as the
input?
Date: Tue, 13 Oct 1998 04:54:23 GMT
From: ritter@io.com (Terry Ritter)
Message-ID: <3622dc88.6129964@news.io.com>
References: <6vrghhroq1@news.umbc.edu>
Newsgroups: sci.crypt
Lines: 89

On 11 Oct 1998 23:55:29 GMT, in <6vrghhroq1@news.umbc.edu>, in
sci.crypt olson@umbc.edu (Bryan G. Olson; CMSC (G)) wrote:

>Terry Ritter (ritter@io.com) wrote:
>
>: My reference is one of the Feistel patents, mid 70's, which an
>: examiner was kind enough to point out as prior art to one of my
>: claims. As far a I know I introduced the idea on sci.crypt several
>: years ago in discussions about my large block designs, You
>: participated in some of those discussions, and some of those are
>: archived on my pages. I am not aware of a text reference.
>
>I'm not sure what specific result you're addressing, but
>I'll once again recommended the paper "Randomized Encryption
>Techniques" by Ronald Rivest and Alan Sherman from Crypto 82.
>Of course it post-dates Feistel, but pre-dates discussions
>on sci.crypt.

Thanks for the reference; until I can get to it, I will assume that it
does indeed cover the homophonic use of a block cipher (the use of a
field in the plaintext block for keying or noise). Presumably it
references the Feistel patent as a prior publication in the field --
if not, the paper must have been rather a hoot for the boys at IBM.

Again -- as far as I know -- I did introduce the homophonic block

cipher construction into a previous discussion on sci.crypt, a
discussion to which you were also a party. Perhaps you would care to
review the archives and tell us exactly what message did present this
construction. I think that message was my own, but I could be wrong.
In either case, who cares? That comment was a reply to the earlier
message, and I think it was appropriate in context.

As far as I remember, the paper mentioned in that discussion of two
years ago was Probabilistic Encryption, 1984, Goldwasser and Micali.
That basically deals with number theoretic homophonic ciphering, but
without using that name, without giving the homophonic block cipher
construction and also without reference to the classic techniques.
That paper is, therefore, different than the technique I was trying to
discuss before the conversation took its negative turn.

But a paper is not a text, and the fact that a 1982 reference has not
made it into the crypto texts should tell us just how well "known"
that reference and this construction really are. (This refers to a
deleted part of my earlier message, which also nobody cares about.)

>[...]
>: Personally, I think "counter mode" is dangerous for just this reason
>: in any Feistel structure, and would always work to avoid the situation
>: where the total plaintext change could be a single increment.
>
>I think the same intuition supports Rivest and Sherman's idea
>of adding randomness by applying an error correction code, and
>then flipping as many bits as the code can correct. Unlike
>the simple random field, it doesn't tell the attacker the location
>of known and unknown plaintext bits.

It seems to me that adding noise to an error-correcting code is in
itself the construction of a homophonic code. It does seem odd to go
to that trouble, only to do it again in a block cipher field. I guess
this would make more sense if they were using the whole plaintext
block, and thus just using the block cipher for strength, but I don't
have the article at hand. Surely we can agree that doing the
necessary error-correction on each deciphering will add overhead to
what otherwise would be a fast process.

As for my intuition, either "counter mode" can be a problem for
Feistel structures, or it cannot:

* If counter mode *is* a problem, there are a lot of proposals around
that recommend it and so need to be changed.

* But if counter mode is *not* a problem, then it is also no problem
for the homophonic construction.

As I see it, the problem with counter mode is that a counting value
produces a strong and potentially unique change signal for each
different bit position. And when the counting field is at the right
side of the block, the bit with the strongest signal is at the edge of
the block, where supposedly "ideal" block cipher properties may be at
their weakest. This is certainly something to avoid in any
construction, including the homophonic block cipher construction.

Terry Ritter ritter@io.com http://www.io.com/~ritter/
Crypto Glossary 1998-08-27: http://www.io.com/~ritter/GLOSSARY.HTM

Subject: Re: Thought question: why is encrypted output routinely the same size as the
input?
Date: 14 Oct 1998 08:47:19 GMT
From: olson@umbc.edu (Bryan G. Olson; CMSC (G))
Message-ID: <701oen$8go$1@news.umbc.edu>
References: <3622dc88.6129964@news.io.com>
Newsgroups: sci.crypt
Lines: 57

Terry Ritter wrote:

: Bryan G. Olson wrote:
[...]
: >I'm not sure what specific result you're addressing, but
: >I'll once again recommended the paper "Randomized Encryption
: >Techniques" by Ronald Rivest and Alan Sherman from Crypto 82.
: >Of course it post-dates Feistel, but pre-dates discussions
: >on sci.crypt.

: Thanks for the reference; until I can get to it, I will assume that it
: does indeed cover the homophonic use of a block cipher (the use of a
: field in the plaintext block for keying or noise). Presumably it
: references the Feistel patent as a prior publication in the field --
: if not, the paper must have been rather a hoot for the boys at IBM.

The "use of a field" is one way to it. The paper cites
30 other works, but not the Feistel patent.

: Again -- as far as I know -- I did introduce the homophonic block
: cipher construction into a previous discussion on sci.crypt, a

: discussion to which you were also a party.

Yes, it's come up several times, and I think you're right
that no one on cares who put it on Usenet first. I usually
point people to Rivest and Sherman because I think it covers
the options nicely. (Maybe I'm biased since Dr. Sherman was
my grad advisor).

: As far as I remember, the paper mentioned in that discussion of two
: years ago was Probabilistic Encryption, 1984, Goldwasser and Micali.
: That basically deals with number theoretic homophonic ciphering, but
: without using that name, without giving the homophonic block cipher
: construction and also without reference to the classic techniques.

Correct. The Rivest and Sherman paper considers both secret
key and public key methods, and references classic ciphers.
Rivest and Sherman cite a 1982 version of the Goldwasser and
Micali paper.

: But a paper is not a text, and the fact that a 1982 reference has not
: made it into the crypto texts should tell us just how well "known"
: that reference and this construction really are.

They get one sentence in the Handbook.

[...]
: * But if counter mode is *not* a problem, then it is also no problem
: for the homophonic construction.

True, but if counter mode is secure, then homophonic block
ciphers are not needed.

--Bryan

Subject: Re: Thought question: why is encrypted output routinely the same size as the
input?
Date: Wed, 14 Oct 1998 15:44:03 GMT
From: ritter@io.com (Terry Ritter)
Message-ID: <3624c694.3976180@news.io.com>
References: <701oen$8go$1@news.umbc.edu>
Newsgroups: sci.crypt
Lines: 73

On 14 Oct 1998 08:47:19 GMT, in <701oen$8go$1@news.umbc.edu>, in
sci.crypt olson@umbc.edu (Bryan G. Olson; CMSC (G)) wrote:

>[...]
>: But a paper is not a text, and the fact that a 1982 reference has not
>: made it into the crypto texts should tell us just how well "known"
>: that reference and this construction really are.
>
>They get one sentence in the Handbook.

Actually there are a couple of references to "randomized encryption"
in the Handbook of Applied Cryptography.

Paragraph 7.3 is a definition of "randomized encryption," which is
basically a homophonic cipher with random keying. There is no use of
the term "homophonic" nor reference to the classical systems or
literature, nor is there any hint of authentication.

Paragraph 8.22 names various ciphers which use randomization. They
give three advantages:

"(i) increasing the effective size of the plaintext message space.
"(ii) precluding or decreasing the effectiveness of chosen-plaintext
attacks by virtue of a one-to-many mapping of plaintext to ciphertext;
and
"(iii) precluding or decreasing the effectiveness of statistical
attacks by leveling the a priori probability distribution of inputs."

But none of these says anything about authentication, and so fail to
describe one of the main opportunities and advantages. One might well
wonder whether the Rivest-Sherman reference is similar.

>[...]
>True, but if counter mode is secure, then homophonic block
>ciphers are not needed.

False. The homophonic construction has more utility than counter
mode:

* Counter mode is dangerous specifically because it is not randomly
keyed. But homophonic mode can be randomly keyed.

* Homophonic mode uses one or more fields in the plaintext block.
This means that The Opponent simply does not have the full plaintext
block to use in known plaintext or defined plaintext attacks.

* Randomized plaintext ciphers (like CBC mode) need other information (like the ciphertext of the previous block) to support deciphering. So when a packet / block is delayed, the next received CBC block can't be immediately deciphered, but instead must wait for delivery. But homophonic blocks can be deciphered independently and immediately.

* Counter mode does not provide authentication, but a homophonic block cipher can be cryptographically self-authenticating. Homophonic mode provides a cryptographic keyed nonlinear error check coding that cannot be reproduced without both the authentication field value and the key.

* The general idea of placing a hidden identifier in each block of plaintext is similar to putting a sending number on a fax, or the ID code that IBM used to have on their copy machines; it can be a way to distinguish different users who have the same key access.

* It is often possible to combine multiple of these advantages (although perhaps not all) in the same homophonic field.

Terry Ritter ritter@io.com http://www.io.com/~ritter/
Crypto Glossary 1998-08-27: http://www.io.com/~ritter/GLOSSARY.HTM

Subject: Re: Thought question: why is encrypted output routinely the same size as the input?
Date: 14 Oct 1998 18:55:39 GMT
From: olson@umbc.edu (Bryan G. Olson; CMSC (G))
Message-ID: <702s3bqld1@news.umbc.edu>
References: <3624c694.3976180@news.io.com>
Newsgroups: sci.crypt
Lines: 55

Terry Ritter (ritter@io.com) wrote:

: On 14 Oct 1998 08:47:19 GMT, in <701oen$8go$1@news.umbc.edu>, in
: sci.crypt olson@umbc.edu (Bryan G. Olson; CMSC (G)) wrote:

: >They get one sentence in the Handbook.

: Actually there are a couple of references to "randomized encryption"
: in the Handbook of Applied Cryptography.

By "they" I meant Rivest and Sherman for the Crypto 82 paper.

: Paragraph 7.3 is a definition of "randomized encryption," which is
: basically a homophonic cipher with random keying. There is no use of
: the term "homophonic" nor reference to the classical systems or
: literature, nor is there any hint of authentication.

: Paragraph 8.22 names various ciphers which use randomization. They
: give three advantages:

: "(i) increasing the effective size of the plaintext message space.
: "(ii) precluding or decreasing the effectiveness of chosen-plaintext
: attacks by virtue of a one-to-many mapping of plaintext to ciphertext;
: and
: "(iii) precluding or decreasing the effectiveness of statistical
: attacks by leveling the a priori probability distribution of inputs."

Those are adapted from Rivest and Sherman.

: But none of these says anything about authentication, and so fail to
: describe one of the main opportunities and advantages. One might well
: wonder whether the Rivest-Sherman reference is similar.

That's because it's the redundancy, not the randomness
that provides authentication. For authentication we need
a cipher in which for any given key the vast majority of
ciphertext blocks do not map to any legal plaintext. That
has nothing to do with homophones.

: >[...]
: >True, but if counter mode is secure, then homophonic block
: >ciphers are not needed.

: False. The homophonic construction has more utility than counter
: mode:

Did you miss the direction of the implication? Randomized
block ciphers are at least as secure as counter mode. The
other advantages you cite are due to adding a redundant
field (except the one that compares to CBC mode instead of
counter mode).

--Bryan

Subject: Re: Thought question: why is encrypted output routinely the same size as the

input?
Date: Thu, 15 Oct 1998 03:31:51 GMT
From: ritter@io.com (Terry Ritter)
Message-ID: <36256c79.6945376@news.io.com>
References: <702s3bqld1@news.umbc.edu>
Newsgroups: sci.crypt
Lines: 67

On 14 Oct 1998 18:55:39 GMT, in <702s3bqld1@news.umbc.edu>, in
sci.crypt olson@umbc.edu (Bryan G. Olson; CMSC (G)) wrote:

>[...]
>: But none of these says anything about authentication, and so fail to
>: describe one of the main opportunities and advantages. One might well
>: wonder whether the Rivest-Sherman reference is similar.
>
>That's because it's the redundancy, not the randomness
>that provides authentication. For authentication we need
>a cipher in which for any given key the vast majority of
>ciphertext blocks do not map to any legal plaintext. That
>has nothing to do with homophones.

On the contrary, that is what homophones do, provided we can label
them and distinguish between them. The data channel is homophonic.
The authentication channel is not.

>: >[...]
>: >True, but if counter mode is secure, then homophonic block
>: >ciphers are not needed.
>
>: False. The homophonic construction has more utility than counter
>: mode:
>
>Did you miss the direction of the implication? Randomized
>block ciphers are at least as secure as counter mode. The
>other advantages you cite are due to adding a redundant
>field (except the one that compares to CBC mode instead of
>counter mode).

That counts.

In starting this off, I responded to the idea of mixing noise with
data. I pointed out that this idea could be extended to do both
authentication and a form of keying. I cited mid-70's Feistel, who
was cited to me by the PTO.

Then I got various learned responses that this was "known." Well,
sure it was, and it was "known" from Feistel. None of the references
cited were before that.

All of which is irrelevant to the real issue, which is that the
homophonic block cipher construction -- essentially "noise" in the
plaintext "data" -- can be advantageous, but nobody really knows about
it. The various popular references *I* have don't say anything about
doing authentication like this. And the texts *I* have don't describe
that.

Is the homophonic block cipher construction presented in Applied
Cryptography? Is it in the Handbook of Applied Cryptography? If not,
the technique is hardly likely to be known by the ordinary reader of
those texts.

I claim there can be real advantages to using the homophonic block
cipher construction, especially when we have large blocks. Surely it
must be hard to disagree with this, yet based on extensive prior
experience, I suspect that you somehow will. But if so, let's see
some quotes.

Terry Ritter ritter@io.com http://www.io.com/~ritter/
Crypto Glossary 1998-08-27: http://www.io.com/~ritter/GLOSSARY.HTM

Subject: Re: Thought question: why is encrypted output routinely the same size as the
input?
Date: Thu, 15 Oct 1998 17:42:51 GMT
From: jsavard@tenMAPSONeerf.edmonton.ab.ca (John Savard)
Message-ID: <3626307e.4493624@news.prosurfr.com>
References: <36256c79.6945376@news.io.com>
Newsgroups: sci.crypt
Lines: 58

ritter@io.com (Terry Ritter) wrote, in part:

>On 14 Oct 1998 18:55:39 GMT, in <702s3bqld1@news.umbc.edu>, in
>sci.crypt olson@umbc.edu (Bryan G. Olson; CMSC (G)) wrote:

>>[...]
>>: But none of these says anything about authentication, and so fail to
>>: describe one of the main opportunities and advantages. One might well
>>: wonder whether the Rivest-Sherman reference is similar.

>>That's because it's the redundancy, not the randomness
>>that provides authentication. For authentication we need
>>a cipher in which for any given key the vast majority of
>>ciphertext blocks do not map to any legal plaintext. That
>>has nothing to do with homophones.

>On the contrary, that is what homophones do, provided we can label
>them and distinguish between them. The data channel is homophonic.
>The authentication channel is not.

>Is the homophonic block cipher construction presented in Applied
>Cryptography? Is it in the Handbook of Applied Cryptography? If not,
>the technique is hardly likely to be known by the ordinary reader of
>those texts.

Applied Cryptography does indeed refer to a system where there are
many collisions in one direction but no collisions in the other
direction, although it may not use the same terminology as you do for
it.

Simple authentication - what Bryan Olson is referring to - involves
redundancy before encryption - i.e., an encrypted checksum of the
message. One could say that if one replaced the checksum by random
bits, one had homophones, but since the wrong checksum combinations
aren't valid, one could indeed quibble about the term being applicable
in that case, and I'd have to agree with him that there is a risk of
causing confusion by using the term here. (OTOH, if the authenticator
were encrypted by an extra level of encryption, and if messages with
invalid authentication were deliberately generated to create confusion
to attackers, then the invalid messages would be homophones at one
level.)

However, what Applied Cryptography was referring to that sounds more
like what you may be talking about was a digital signature method in
which collisions of one type made it likely that a forgery would be
detected by denying the information needed for an attack. I shall have
to look at my copy this evening to refresh my memory here.

AC doesn't mention, however, the idea of adding a few extra bits to
every block, either for variation (like an IV) or for authentication.
These techniques are only discussed on a per-message basis (of course,
a communications block, say 1024 bytes, could be handled as a message,
but the idea of doing it for each 64-bit or 128-bit encryption block
is definitely not noted).

John Savard
http://members.xoom.com/quadibloc/index.html

Subject: Re: Thought question: why is encrypted output routinely the same size as the
input?
Date: Sat, 17 Oct 1998 17:20:10 GMT
From: ritter@io.com (Terry Ritter)
Message-ID: <3628d1c6.6287748@news.io.com>
References: <3626307e.4493624@news.prosurfr.com>
Newsgroups: sci.crypt
Lines: 126

On Thu, 15 Oct 1998 17:42:51 GMT, in
<3626307e.4493624@news.prosurfr.com>, in sci.crypt
jsavard@tenMAPSONeerf.edmonton.ab.ca (John Savard) wrote:

>[...]
>>Is the homophonic block cipher construction presented in Applied
>>Cryptography? Is it in the Handbook of Applied Cryptography? If not,
>>the technique is hardly likely to be known by the ordinary reader of
>>those texts.
>
>Applied Cryptography does indeed refer to a system where there are
>many collisions in one direction but no collisions in the other
>direction, although it may not use the same terminology as you do for
>it.

So the never-ending argument continues about whether or not the
technique of adding "extra information" to the plaintext block of a
block cipher for keying and authentication is well known. Jeez, I
guess we could just take a poll.

Did *you* know that one could authenticate individual blocks in
exactly this way? If so, where did you learn it? (Certainly I have
discussed this here before, and described it also on my pages and my
Crypto Glossary.)

This technique has little to do with a "system," but instead is
confined to individual blocks. That is the meaning of block
independence, which is one of the advantages of the technique. And
the technique is most useful with *large* blocks, which is one of the
advantages of having *large* blocks.

>Simple authentication - what Bryan Olson is referring to - involves

>redundancy before encryption - i.e., an encrypted checksum of the
>message. One could say that if one replaced the checksum by random
>bits, one had homophones, but since the wrong checksum combinations
>aren't valid,

I believe I pointed out previously that the error-correcting code with errors example was indeed a homophonic coding. But it is not a *secret* coding; it is not a homophonic cipher.

But we can accomplish the same end *without* using an error-correcting code. Which means we have an advance, folks.

>one could indeed quibble about the term being applicable
>in that case, and I'd have to agree with him that there is a risk of
>causing confusion by using the term here.

Oh, please. The only real confusion here is that some people cannot stand their crypto background or favorite books to be shown up as not including a simple technique of significant value, and they are willing to confuse and distort the conversation to hide that bitter truth. If that were not the case, we would see direct quotes that we could all evaluate -- but we don't. Or we might see arguments confined to the worth of the technique (instead of whether we all know about it) -- but we don't. We might see issues formulated to be clear and testable so they could actually be resolved -- but we don't. There is nothing new about this. It occurs all the time on sci.crypt. It really is very embarrassing.

The reason I sometimes use the same term for these fields is because exactly the same process occurs. To the cipher there is no difference between any of these fields. When one labels these fields as having a particular purpose, one is simply labeling one's own interpretations. And while that can be useful, believing the joke and demanding that everyone else recognize it is something else again.

>(OTOH, if the authenticator
>were encrypted by an extra level of encryption, and if messages with
>invalid authentication were deliberately generated to create confusion
>to attackers, then the invalid messages would be homophones at one
>level.)

Simply having the ability to select from among a wide array of ciphertext code values for a particular plaintext is homophonic. Please feel free to see:

http://www.io.com/~ritter/GLOSSARY.HTM#Homophonic
http://www.io.com/~ritter/GLOSSARY.HTM#HomophonicSubstitution

Does a classical homophonic cipher not become homophonic until the first homophone goes out?

When we confine the input data to part of a block we get the homophonic effect. There is very little here to discuss about agreeing or not agreeing or seeing things one way or the other. This construction creates homophonic blocks from normal block ciphers, it can be useful, and it is generally unknown. If you disagree, give quotes.

>However, what Applied Cryptography was referring to that sounds more
>like what you may be talking about was a digital signature method in
>which collisions of one type made it likely that a forgery would be
>detected by denying the information needed for an attack. I shall have
>to look at my copy this evening to refresh my memory here.
>
>AC doesn't mention, however, the idea of adding a few extra bits to
>every block, either for variation (like an IV) or for authentication.

Since this last comment *is* what I described (except for the "few" part), I fail to see the point of your previous comments. If the technique is not known in common texts, it is not well known.

My point always has been that this useful and valuable technique is *not* well known, and is *not* in the common texts. Please address the point. Include quotes.

>These techniques are only discussed on a per-message basis (of course,
>a communications block, say 1024 bytes, could be handled as a message,
>but the idea of doing it for each 64-bit or 128-bit encryption block
>is definitely not noted).

And, if those techniques *are* only discussed on a per-message basis, they are obviously *not* the technique of this discussion. So your point would be?

Terry Ritter ritter@io.com http://www.io.com/~ritter/
Crypto Glossary http://www.io.com/~ritter/GLOSSARY.HTM

Subject: Re: Thought question: why is encrypted output routinely the same size as the

input?
Date: Sun, 18 Oct 1998 00:14:28 -0600
From: jgfunj@EnqvbSerrGrknf.pbz (W T Shaw)
Message-ID: <jgfunj-1810980014280001@207.22.198.205>
References: <3628d1c6.6287748@news.io.com>
Newsgroups: sci.crypt
Lines: 25

In article <3628d1c6.6287748@news.io.com>, ritter@io.com (Terry Ritter) wrote:

> On Thu, 15 Oct 1998 17:42:51 GMT, in
> <3626307e.4493624@news.prosurfr.com>, in sci.crypt
> jsavard@tenMAPSONeerf.edmonton.ab.ca (John Savard) wrote:
>
> >These techniques are only discussed on a per-message basis (of course,
> >a communications block, say 1024 bytes, could be handled as a message,
> >but the idea of doing it for each 64-bit or 128-bit encryption block
> >is definitely not noted).
>
> And, if those techniques *are* only discussed on a per-message basis,
> they are obviously *not* the technique of this discussion.

I read several places where a specified signature length is considered.
There need not be any particular size. Indeed, the MAC's that I generate
are additive; I seamlessly add the results from sequential blocks
together. One could hash this down to any length, but that is not
particularily helpful in finding where a corruption of the original might
have occured, which I would consider most important in some cases.
--

Insanity means doing the same thing over and over again and expecting different
results...like CDA2.

Decrypt with ROT13 to get correct email address.

Subject: Re: Thought question: why is encrypted output routinely the same size as the
input?
Date: 17 Oct 1998 08:53:11 GMT
From: olson@umbc.edu (Bryan G. Olson; CMSC (G))
Message-ID: <709ltn$g09$1@news.umbc.edu>
References: <36256c79.6945376@news.io.com>
Newsgroups: sci.crypt
Lines: 72

Terry Ritter (ritter@io.com) wrote:

: On 14 Oct 1998 18:55:39 GMT, in <702s3bqld1@news.umbc.edu>, in
: sci.crypt olson@umbc.edu (Bryan G. Olson; CMSC (G)) wrote:

: >[...]
: >: But none of these says anything about authentication, and so fail to
: >: describe one of the main opportunities and advantages. One might well
: >: wonder whether the Rivest-Sherman reference is similar.
: >
: >That's because it's the redundancy, not the randomness
: >that provides authentication. For authentication we need
: >a cipher in which for any given key the vast majority of
: >ciphertext blocks do not map to any legal plaintext. That
: >has nothing to do with homophones.

: On the contrary, that is what homophones do, provided we can label
: them and distinguish between them. The data channel is homophonic.
: The authentication channel is not.

Cryptologists talk about the sender encrypting a plaintext
into a ciphertext which gets transmitted to the receiver.
You seem to have this two channel model. Where did you get
it? Why do you need it? What do think "channel" means?

Homophones are multiple ciphertexts induced by the same
key and the same plaintext. That's what they are; what
they do is increase the entropy of the ciphertext. They do
not provide authentication, since each homophone has a
corresponding legitimate plaintext.

Suppose we expand a plaintext block by n bits. Say the
n bit expansion adds k bits of entropy. Then on average
each plaintext has 2^k homophones. A random text has a
probability of 1/2^(n-k) of corresponding to some expanded
plaintext, and we can use this for authentication by
rejecting all those that don't decrypt to a possible block.
Under the random cipher model an adversary has the
1/2^(n-k) chance of successful forgery on each block.

That n-k number is the measure of redundancy. The number
of truly random bits doesn't effect the chance at all.
That qualification "provided we can label them and
distinguish between them" hides what's really going on. You
have to add redundancy to get authentication, and then you
can throw out the randomness and the authentication doesn't
go away.

That's not to say the randomness is worthless - it does just

what Rivest and Sherman said it does. In practice our ciphers
are not random permutations, so the advantages cited may
increase security of both privacy and authentication.

[...]
: I claim there can be real advantages to using the homophonic block
: cipher construction, especially when we have large blocks. Surely it
: must be hard to disagree with this, yet based on extensive prior
: experience, I suspect that you somehow will. But if so, let's see
: some quotes.

What are you talking about? I've been recommending the Rivest
and Sherman paper on this group for years. It's the major
paper advocating the technique and analyzing the advantages.
As Rivest and Sherman state "The goal of randomized encryption
is increased security". That security may apply to both
privacy and authentication. But homophones do not provide a
means to authenticate. Redundancy, not randomness, offers
authentication.

--Bryan

Subject: Re: Thought question: why is encrypted output routinely the same size as the
input?
Date: Sat, 17 Oct 1998 17:20:20 GMT
From: ritter@io.com (Terry Ritter)
Message-ID: <3628d1cd.6295356@news.io.com>
References: <709ltn$g09$1@news.umbc.edu>
Newsgroups: sci.crypt
Lines: 172

On 17 Oct 1998 08:53:11 GMT, in <709ltn$g09$1@news.umbc.edu>, in
sci.crypt olson@umbc.edu (Bryan G. Olson; CMSC (G)) wrote:

>Terry Ritter (ritter@io.com) wrote:
>
>: On 14 Oct 1998 18:55:39 GMT, in <702s3bqld1@news.umbc.edu>, in
>: sci.crypt olson@umbc.edu (Bryan G. Olson; CMSC (G)) wrote:
>
>: >[...]
>: >: But none of these says anything about authentication, and so fail to
>: >: describe one of the main opportunities and advantages. One might well
>: >: wonder whether the Rivest-Sherman reference is similar.
>: >
>: >That's because it's the redundancy, not the randomness
>: >that provides authentication. For authentication we need
>: >a cipher in which for any given key the vast majority of
>: >ciphertext blocks do not map to any legal plaintext. That
>: >has nothing to do with homophones.
>
>: On the contrary, that is what homophones do, provided we can label
>: them and distinguish between them. The data channel is homophonic.
>: The authentication channel is not.
>
>Cryptologists talk about the sender encrypting a plaintext
>into a ciphertext which gets transmitted to the receiver.
>You seem to have this two channel model. Where did you get
>it? Why do you need it? What do think "channel" means?

Electrical Engineers have some modest acquaintance with the term
"channel." Perhaps you should expand your view.

In general, a "channel" is bandwidth reserved for a particular
connection or information type. Here we take the second tack. For
example, some systems use a "control channel" along with a "data
channel" to control data flow over the same connection.

When we partition the plaintext block into a section for message data
and one or more other sections for authentication data and keying
data, each of these sections can be considered a separate "channel."

>Homophones are multiple ciphertexts induced by the same
>key and the same plaintext. That's what they are; what
>they do is increase the entropy of the ciphertext. They do
>not provide authentication, since each homophone has a
>corresponding legitimate plaintext.

I can agree with the definition. But you seem to have gone quite a
ways beyond that.

The homophonic block cipher constructs homophonic ciphertext simply by
partitioning the plaintext block into message and keying fields (or
channels). Each different value in the keying field generates a
different ciphertext for exactly the same data in the message field.
This is homophonic operation, by your definition.

In classical cryptography, homophonic systems may not distinguish
between homophones at all. And all that means is that the modern-day
homophonic block cipher construction goes beyond the classical

designs, just as all modern systems do. Your definition says nothing about an inability to distinguish between homophones, and if it did, that would define a type of homophone system, not homophones themselves.

>Suppose we expand a plaintext block by n bits. Say the
>n bit expansion adds k bits of entropy. Then on average
>each plaintext has 2^k homophones. A random text has a
>probability of 1/2^(n-k) of corresponding to some expanded
>plaintext, and we can use this for authentication by
>rejecting all those that don't decrypt to a possible block.
>Under the random cipher model an adversary has the
>1/2^(n-k) chance of successful forgery on each block.

I accept this. I was just going to explain it to you.

>That n-k number is the measure of redundancy. The number
>of truly random bits doesn't effect the chance at all.
>That qualification "provided we can label them and
>distinguish between them" hides what's really going on. You
>have to add redundancy to get authentication, and then you
>can throw out the randomness and the authentication doesn't
>go away.

Generally right.

So what a surprise that -- as I have been saying all along -- the homophonic block cipher construction is mainly useful in ciphers with large blocks, in which we can afford some overhead.

>That's not to say the randomness is worthless - it does just
>what Rivest and Sherman said it does.

The source of this idea was Feistel, years earlier. It just does what Feistel said it does.

>In practice our ciphers
>are not random permutations, so the advantages cited may
>increase security of both privacy and authentication.
>
>[...]
>: I claim there can be real advantages to using the homophonic block
>: cipher construction, especially when we have large blocks. Surely it
>: must be hard to disagree with this, yet based on extensive prior
>: experience, I suspect that you somehow will. But if so, let's see
>: some quotes.
>
>What are you talking about? I've been recommending the Rivest
>and Sherman paper on this group for years.

Quote please.

>It's the major
>paper advocating the technique and analyzing the advantages.
>As Rivest and Sherman state "The goal of randomized encryption
>is increased security". That security may apply to both
>privacy and authentication.

Quote please.

>But homophones do not provide a
>means to authenticate. Redundancy, not randomness, offers
>authentication.

Try to follow along:

1) The homophonic block cipher construction partitions the plaintext into multiple "fields" or "communications sub-channels."

2) One field is used for message data.

3) Another field can be used for keying data. Each value in this field creates a different ciphertext for exactly the same message data. Each of these ciphertexts *is* a homophonic code for the same message. In a sense, each of these homophonic ciphertexts is "numbered" by the keying field. When we decipher, we can throw that number away if we so decide. This is classical homophonic cryptography.

4) Yet another field can be used for authentication. Each value in this field *also* creates a different ciphertext for exactly the same message data. Each of these ciphertext is *also* a homophonic code for the same message. But, since the homophones are numbered, we *can* distinguish between them if we wish. We can thus identify the "correct" homophone, by whatever way we wish to identify a correct authentication value. This process is analogous to other forms of error detection used for authentication like CRC, but CRC is weak, whereas the homophonic form can be strong cryptographic authentication.

Now, each of these different fields may be *used* in different ways.
Yet they have *exactly* *the* *same* *construction*. Whatever words
you want to use for it, the same thing is happening in every field.
Homophones are created and distinguished in two fields, both of which
are treated by the cipher in exactly the same way. The only thing
that changes is our *interpretation* of the field, and if you wish to
put two different names on that same operation, well, fine. Certainly
I call them "keying" and "authentication." If you want to call them
"randomness" and "redundancy," feel free. But don't imagine that you
have the only two terms that can be properly used.

Terry Ritter ritter@io.com http://www.io.com/~ritter/
Crypto Glossary http://www.io.com/~ritter/GLOSSARY.HTM

Subject: Re: Thought question: why is encrypted output routinely the same size as the input?
Date: 18 Oct 1998 01:21:19 GMT
From: olson@umbc.edu (Bryan G. Olson; CMSC (G))
Message-ID: <70bfqf$3om$1@news.umbc.edu>
References: <3628d1cd.6295356@news.io.com>
Newsgroups: sci.crypt
Lines: 170

Terry Ritter (ritter@io.com) wrote:

: Bryan G. Olson; CMSC (G)) wrote:

: >Terry Ritter (ritter@io.com) wrote:
: >
: >: Bryan G. Olson wrote:
: >: >it's the redundancy, not the randomness
: >: >that provides authentication. For authentication we need
: >: >a cipher in which for any given key the vast majority of
: >: >ciphertext blocks do not map to any legal plaintext. That
: >: >has nothing to do with homophones.
: >
: >: On the contrary, that is what homophones do, provided we can label
: >: them and distinguish between them. The data channel is homophonic.
: >: The authentication channel is not.
[...]

: >Homophones are multiple ciphertexts induced by the same
: >key and the same plaintext. That's what they are; what
: >they do is increase the entropy of the ciphertext. They do
: >not provide authentication, since each homophone has a
: >corresponding legitimate plaintext.

: I can agree with the definition. But you seem to have gone quite a
: ways beyond that.

Well, that's something.

: The homophonic block cipher constructs homophonic ciphertext simply by
: partitioning the plaintext block into message and keying fields (or
: channels). Each different value in the keying field generates a
: different ciphertext for exactly the same data in the message field.
: This is homophonic operation, by your definition.

Absolutely, though it "A", not "The" homophonic block cipher construction.

: In classical cryptography, homophonic systems may not distinguish
: between homophones at all. And all that means is that the modern-day
: homophonic block cipher construction goes beyond the classical
: designs, just as all modern systems do. Your definition says nothing
: about an inability to distinguish between homophones, and if it did,
: that would define a type of homophone system, not homophones
: themselves.

: >Suppose we expand a plaintext block by n bits. Say the
: >n bit expansion adds k bits of entropy. Then on average
: >each plaintext has 2^k homophones. A random text has a
: >probability of 1/2^(n-k) of corresponding to some expanded
: >plaintext, and we can use this for authentication by
: >rejecting all those that don't decrypt to a possible block.
: >Under the random cipher model an adversary has the
: >1/2^(n-k) chance of successful forgery on each block.

: I accept this. I was just going to explain it to you.

: >That n-k number is the measure of redundancy. The number
: >of truly random bits doesn't effect the chance at all.
: >That qualification "provided we can label them and
: >distinguish between them" hides what's really going on. You
: >have to add redundancy to get authentication, and then you
: >can throw out the randomness and the authentication doesn't
: >go away.

: Generally right.

: So what a surprise that -- as I have been saying all along -- the
: homophonic block cipher construction is mainly useful in ciphers with
: large blocks, in which we can afford some overhead.

This is a surprise why?

: >: Surely it
: >: must be hard to disagree with this, yet based on extensive prior
: >: experience, I suspect that you somehow will. But if so, let's see
: >: some quotes.
: >
: >What are you talking about? I've been recommending the Rivest
: >and Sherman paper on this group for years.

: Quote please.

Quote of what? I have to document that I don't disagree?

[...]
: Try to follow along:

O.K. I read it and I think I followed. I'll intersperse
my comments.

: 1) The homophonic block cipher construction partitions the plaintext
: into multiple "fields" or "communications sub-channels."

We agreed on what homophonic means. All (invertible)
homophonic block ciphers expand the ciphertext, but they
may or may not do so by adding fields to the plaintext
before encryption.

All constructions that add fields to the plaintext before
encryption also expand the ciphertext. They may or may
not be homophonic. Specifically, they are homophonic if
and only if for a given key and plaintext, the construction
admits more than one possible field value (or if it was
already homophonic without the extra field).

: 2) One field is used for message data.

I follow, but conventional terminology is that the message data
_is_ the plaintext. See Applied Cryptography page 1, or the
Handbook page 11. Fields added by the cryptographic process
are not properly plaintext.

: 3) Another field can be used for keying data.

And that gives us _one_ way to produce a homophonic block cipher.
It is not synonymous with homophonic block ciphers in general.

: Each value in this
: field creates a different ciphertext for exactly the same message
: data. Each of these ciphertexts *is* a homophonic code for the same
: message. In a sense, each of these homophonic ciphertexts is
: "numbered" by the keying field. When we decipher, we can throw that
: number away if we so decide. This is classical homophonic
: cryptography.

Agreed.

: 4) Yet another field can be used for authentication. Each value in
: this field *also* creates a different ciphertext for exactly the same
: message data.

If, only if, and to the degree that the content of the field
is nondeterministic. If your construction always puts the same
value in that field for the same values elsewhere, then it
cannot induce homophones.

: Each of these ciphertext is *also* a homophonic code
: for the same message. But, since the homophones are numbered, we
: *can* distinguish between them if we wish. We can thus identify the
: "correct" homophone, by whatever way we wish to identify a correct
: authentication value. This process is analogous to other forms of
: error detection used for authentication like CRC, but CRC is weak,
: whereas the homophonic form can be strong cryptographic
: authentication.

: Now, each of these different fields may be *used* in different ways.
: Yet they have *exactly* *the* *same* *construction*.

But you've confused the extra field construction with
"homophonic". You can make the cipher homophonic by putting
in an extra random field. You can use an extra field for
authentication by filling it with redundant data (whether or
not there's another nondeterministic expansion). That doesn't
mean extra redundant fields are homophonic.

: Whatever words
: you want to use for it, the same thing is happening in every field.

The issue is what goes into the field, not what we call it.

If it's random - additional entropy - nondeterministic, that
means it induces homophones. If it's redundant - deterministic,
then and only then is it useful for authentication.

--Bryan

Subject: Re: Thought question: why is encrypted output routinely the same size as the
input?
Date: Sun, 18 Oct 1998 00:02:30 -0600
From: jgfunj@EnqvbSerrGrknf.pbz (W T Shaw)
Message-ID: <jgfunj-1810980002300001@207.22.198.205>
References: <709ltn$g09$1@news.umbc.edu>
Newsgroups: sci.crypt
Lines: 102

In article <709ltn$g09$1@news.umbc.edu>, olson@umbc.edu (Bryan G. Olson;
CMSC (G)) wrote:
>
> Homophones are multiple ciphertexts induced by the same
> key and the same plaintext. That's what they are; what
> they do is increase the entropy of the ciphertext. They do
> not provide authentication, since each homophone has a
> corresponding legitimate plaintext.

Actually, there is a need for more information, some source to choose
amongst the possible outputs. (This can also be recovered intact in
decryption. In fact, the nature of the choices can amount to a second
channel, even being the main channel itself with the rest representing
noise.)

The selections can be driven more or less by an input of randomness, a
picking of alternatives beyond user definition.

If the nature of these selections is recovered, it can be used in the
authenticaton process to modify the results, each homophone could produce
a different plaintext, even with the same key, legitimacy is merely a
subjective evaluation of the results of a deductive process.
>
> Suppose we expand a plaintext block by n bits. Say the
> n bit expansion adds k bits of entropy. Then on average
> each plaintext has 2^k homophones. A random text has a
> probability of 1/2^(n-k) of corresponding to some expanded
> plaintext, and we can use this for authentication by
> rejecting all those that don't decrypt to a possible block.
> Under the random cipher model an adversary has the
> 1/2^(n-k) chance of successful forgery on each block.
>
> That n-k number is the measure of redundancy. The number
> of truly random bits doesn't effect the chance at all.
> That qualification "provided we can label them and
> distinguish between them" hides what's really going on. You
> have to add redundancy to get authentication, and then you
> can throw out the randomness and the authentication doesn't
> go away.

You need not add the redundancy to get authentication, merely treat the
string as if it had had it added.

Using an algorithm especially tuned to demonstrate this, taking the above
sentence, I will encrypt it, adding randomness, 2^54. I will use the same
key to get a MAC from both the encrypted text, and the plaintext:

Encrypted:
{zqfz*,7*gn\m3,h(_^r4<8min@9)c_z9w+0g-avp!e7yep*@*g\
2@o$17?v|;iye=)(qy4r`b\&[3er_#atq@0kt`<!&e|-d8y^#zd qa(__n0s.ce>t~ }

MAC from Encrypted:
`you|need|not|add|the|redundancy|to|get|authentication,|mere
ly|treat|the|string|as|if|it|had|had|it|added.|

MAC from Plaintext:
su)/]+,>0j%.*h,;iz)w$/q)8_5o6e%?f>vy"y,i!vv^$>qu=mjea]+yh|'?
2d%!kc7x6o#|7i*`txsk4@(a,j"nh6%/@b6%8|

Double MAC, MAC of above MAC:
zk*[@;8@4)]^jnp$^*+pf072*4$-i,t6d;&n()emg-d#)"%e9!<4,-vv-0/%
4]o&+d)^&;0"y!x<<t-'#n[i(d$+x5|

All homophonic encryptions using the same user defined keys would produce
the same MAC appearing as recognizable text, while a simple change in the
keys would make them all different. I say *keys* because with the
cryptosystem at hand, it takes two; hereinafter below, I will consider an
immediate combination as a single key.

A MAC based on plaintext is shorter than the plaintext by the same amount
as the normal encryption of the text would be longer than the plaintext.
The Double MAC is twice a shorter. There may be even more possibilities
than I show.

To get the same length of MAC as plaintext, encrypt with one key and

generate a MAC from the cyphertext with another. Since versions of the encryptions are all going to be different, one of 2^54 in this case, changing the key can cause a similiar quantity of different MAC's that you can get. A different keyed MAC would be only useful to authenticate a particular encryption since for all practical purposes, this is a one-way process.

> ...As Rivest and Sherman state "The goal of randomized encryption
> is increased security". That security may apply to both
> privacy and authentication. But homophones do not provide a
> means to authenticate. Redundancy, not randomness, offers
> authentication.
>
I believe that the example I gave demonstrates that it can. In the system used, a private key, known only to a single user, could be used to render a MAC of any plaintext or ciphertext, even another MAC made by another key. The requirement to make this work unconditionally is in using different keys.

It is obvious to me that an algorithm that produces output variability can use the same characteristic quite easily in authentication. It seemed natural to me to combine them, which I have done.
--

Insanity means doing the same thing over and over again and expecting different results...like CDA2.

Decrypt with ROT13 to get correct email address.

Subject: Re: Thought question: why is encrypted output routinely the same size as the input?
Date: Wed, 14 Oct 1998 23:46:32 -0600
From: jgfunj@EnqvbSerrGrknf.pbz (W T Shaw)
Message-ID: <jgfunj-1410982346320001@dialup115.itexas.net>
References: <702s3bqld1@news.umbc.edu>
Newsgroups: sci.crypt
Lines: 20

In article <702s3bqld1@news.umbc.edu>, olson@umbc.edu (Bryan G. Olson; CMSC (G)) wrote:
>
> That's because it's the redundancy, not the randomness
> that provides authentication. For authentication we need
> a cipher in which for any given key the vast majority of
> ciphertext blocks do not map to any legal plaintext. That
> has nothing to do with homophones.
>
Strength of an algorithm in encryption can be related to strength of an algorithm in producing a MAC. In authentication using a MAC, the mapping can be to only one plaintext, jumbled garbage if ciphertext is used as the input and a wrong key is used to get output, or if plaintext is used as input and any key is used to find an output.

In short the more output possibilities in encryptive mode, the stronger it is in MAC mode.
--

Decrypt with ROT13 to get correct email address.

Subject: Re: Thought question: why is encrypted output routinely the same size as the input?
Date: Mon, 12 Oct 1998 14:25:26 +0100
From: Mok-Kong Shen <mok-kong.shen@stud.uni-muenchen.de>
Message-ID: <36220346.9B039373@stud.uni-muenchen.de>
References: <361a3f1a.5668671@news.prosurfr.com>
Newsgroups: sci.crypt
Lines: 22

John Savard wrote:

> However, I could use a different mode that does eliminate a direct
> known-plaintext attack on DES, and yet is still insecure:
>
> send, enciphered, a random 64-bit block, then XOR that block to every
> block in the message before encryption.
>
> I don't know, now, the value of any plaintext block. But I can still
> brute-force search: decipher a *pair* of blocks for every key, and
> when the difference between them, deciphered, is the same as that
> between the two plaintext blocks, I probably have the right key.
>
> This is a trivial example, but it illustrates the principle that a
> method that _apparently_ eliminates a true known-plaintext attack may
> not actually give security to a cipher, even if that cipher is not
> vulnerable in the case of unknown plaintext.

If the the random 64-bit block is encrypted with another key, wouldn't that be a good scheme?

M. K. Shen

Subject: Re: Thought question: why is encrypted output routinely the same size as the input?
Date: Mon, 12 Oct 1998 14:57:21 +0100
From: Mok-Kong Shen <mok-kong.shen@stud.uni-muenchen.de>
Message-ID: <36220AC1.AEFFD28E@stud.uni-muenchen.de>
References: <36190b22.6139167@news.io.com>
Newsgroups: sci.crypt
Lines: 32

Terry Ritter wrote:

> Obviously I have been unable to communicate what the a/k stuff is.
>
> Maybe:
>
>   C[i] = E(k, (P[i] << 64) + a/k[i]))
>
> where E is a 64 byte block cipher
>   C[i] is 64 byte ciphertext
>   P[i] is 56 byte plaintext
>   a/k[i] is 8 byte (64 bit) authentication / keying value
>
> We can assume that every possible value for a/k will produce a
> different ciphertext, and that every ciphertext bit will have an equal
> opportunity to change.
>
> This means there will be a multiplicity of ciphertext representations
> for the exact same data: this is a homophonic block cipher. And that

This is certainly a convenient way of obtaining homophones (of 56 byte) entities. I think another way would be this: Expand each byte (8 bit entities) to 9 bit homophones. This gives 63 bytes. The 64th byte could be random. I guess this would be more difficult to analyse (though it does not provide the 'authentication').

In a previous post you mentioned the advantages of large blocks. I agree with you on the desirability of larger block sizes but wonder why it has apparently not been stressed in the literature as it deserves. (Because of hardware implementations?)

M. K. Shen

Subject: Re: Thought question: why is encrypted output routinely the same size as the input?
Date: Mon, 12 Oct 1998 18:20:34 GMT
From: ritter@io.com (Terry Ritter)
Message-ID: <36224706.5167449@news.io.com>
References: <36220AC1.AEFFD28E@stud.uni-muenchen.de>
Newsgroups: sci.crypt
Lines: 55

On Mon, 12 Oct 1998 14:57:21 +0100, in <36220AC1.AEFFD28E@stud.uni-muenchen.de>, in sci.crypt Mok-Kong Shen <mok-kong.shen@stud.uni-muenchen.de> wrote:

>[...]
>This is certainly a convenient way of obtaining homophones (of 56
>byte) entities. I think another way would be this: Expand each byte
>(8 bit entities) to 9 bit homophones. This gives 63 bytes. The 64th
>byte could be random. I guess this would be more difficult to
>analyse (though it does not provide the 'authentication').

Sure. Some benefit flows from simply having more data space than message space. But if we don't collect the extra space in one place, it will be more difficult to both set and check, and we may be more tempted to say that the extra space is not worth the trouble.

>In a previous post you mentioned the advantages of large blocks.
>I agree with you on the desirability of larger block sizes but
>wonder why it has apparently not been stressed in the literature as
>it deserves. (Because of hardware implementations?)

My guess is that the avoidance of large blocks is first related to the use of Feistel constructions for block ciphers. Feistel constructions need a half-block-wide f() function, which ideally is 1:1 (a permutation) as in DES. It is just difficult to construct ciphers with large blocks using "normal" technology.

Even here on sci.crypt I have often seen the opinion expressed that a large block cipher simply could not provide the necessary mixing or diffusion. The presentation of FFT-style mixing with log n levels is at least a plausible counterexample, but it has not appeared in the texts.

Another problem is a general lack of knowledge of the advantages of large blocks. When you have a hammer, you tend to see all your problems as nails, and here people have a particular vision of a DES-like block cipher. Changing that view requires changing many other things about the way ciphers operate and which had been thought fully resolved. Some of the advantages of large blocks are best

realized by system re-design, as opposed to simply dropping a new implementation into the DES-box slot.

I suppose tradeoffs are inherently more engineering than general academics, but we don't have a good engineering text on crypto, and the academic texts have uniformly failed to explore the possibilities of new structures and new designs. Perhaps after AES we will see texts with wider horizons.

Terry Ritter ritter@io.com http://www.io.com/~ritter/
Crypto Glossary 1998-08-27: http://www.io.com/~ritter/GLOSSARY.HTM

Subject: Re: Thought question: why is encrypted output routinely the same size as the input?
Date: 13 Oct 1998 09:35:48 +0100
From: Paul Crowley <paul@hedonism.demon.co.uk>
Message-ID: <8767do3llp.fsf@hedonism.demon.co.uk>
References: <36224706.5167449@news.io.com>
Newsgroups: sci.crypt
Lines: 34

ritter@io.com (Terry Ritter) writes:
> My guess is that the avoidance of large blocks is first related to the
> use of Feistel constructions for block ciphers. Feistel constructions
> need a half-block-wide f() function, which ideally is 1:1 (a
> permutation) as in DES. It is just difficult to construct ciphers
> with large blocks using "normal" technology.

I'm in the process of designing a very fast large block cipher based around a balanced Feistel network, and my current favourite design does use a bijective F function. I've found the use of a Feistel construction has made the cipher easier to think about, and I'd be interested to know what disadvantages you see.

http://www.hedonism.demon.co.uk/paul/mercy

> Even here on sci.crypt I have often seen the opinion expressed that a
> large block cipher simply could not provide the necessary mixing or
> diffusion. The presentation of FFT-style mixing with log n levels is
> at least a plausible counterexample, but it has not appeared in the
> texts.

One design I've played with uses an F function with data-dependent indexing into the right half; this isn't used in the cipher at the moment, but does appear in the current revision of the key schedule. I suspect that with work such a construction could provide diffusion as rapidly as the rotates in RC5; however, it pretty much rules out a bijective F-function so far as I can tell. You even could shuffle the words in the left half using randomising information taken from the right half, though I suspect that data-dependent indexing provides the same advantages.
--
__
\/ o\ paul@hedonism.demon.co.uk Edinburgh fetish club Permission \ /
/__/ Paul Crowley Nov 8 http://www.hedonism.demon.co.uk/permission /~\

Subject: Re: Thought question: why is encrypted output routinely the same size as the input?
Date: Tue, 13 Oct 1998 15:31:39 +0100
From: Mok-Kong Shen <mok-kong.shen@stud.uni-muenchen.de>
Message-ID: <3623644B.167B2050@stud.uni-muenchen.de>
References: <8767do3llp.fsf@hedonism.demon.co.uk>
Newsgroups: sci.crypt
Lines: 28

Paul Crowley wrote:

> http://www.hedonism.demon.co.uk/paul/mercy

I only very quickly went through your Web page. Naive questions: What does you check in 'key validation'? Do you mean the key has to consist of characters in the English alphabet and nothing else? I suppose that the task of turning a passphrase into a key is because the system can accepts only a key of a fixed format. Is this right?

> One design I've played with uses an F function with data-dependent
> indexing into the right half; this isn't used in the cipher at the
> moment, but does appear in the current revision of the key schedule.
< I suspect that with work such a construction could provide diffusion
< as rapidly as the rotates in RC5; however, it pretty much rules out a
< bijective F-function so far as I can tell. You even could shuffle the

I believe that this is advantageous. This comes from my experience with a design of my own where I chain blocks using some hash value of a previous block to modify the encryption process of the current block. There is some parallels in the ideas.

I don't understand your 'it pretty much rules out a bijective

F-function'. Doesn't the possibility of construction depend on the
the type of constructin that you invisage?

M. K. Shen

Subject: Re: Thought question: why is encrypted output routinely the same size as the input?
Date: Tue, 13 Oct 1998 15:19:17 GMT
From: ritter@io.com (Terry Ritter)
Message-ID: <36236dfe.5386998@news.io.com>
References: <8767do3llp.fsf@hedonism.demon.co.uk>
Newsgroups: sci.crypt
Lines: 104

On 13 Oct 1998 09:35:48 +0100, in
<8767do3llp.fsf@hedonism.demon.co.uk>, in sci.crypt Paul Crowley
<paul@hedonism.demon.co.uk> wrote:

>ritter@io.com (Terry Ritter) writes:
>> My guess is that the avoidance of large blocks is first related to the
>> use of Feistel constructions for block ciphers. Feistel constructions
>> need a half-block-wide f() function, which ideally is 1:1 (a
>> permutation) as in DES. It is just difficult to construct ciphers
>> with large blocks using "normal" technology.
>
>I'm in the process of designing a very fast large block cipher based
>around a balanced Feistel network, and my current favourite design
>does use a bijective F function. I've found the use of a Feistel
>construction has made the cipher easier to think about,

Easier than what? To what are you comparing it, and why should
Feistel be easier to think of than, say, FFT-style mixings? In
particular, why should a complex design with many different component
types arranged in irregular ways be easier to think about than a
design with few component types arranged in regular ways?

Suppose we have a 1 bit change in the input: Feistel must propagate
that change through a complex f() with properties which do not
directly apply to reasoning about this case, or diffusion in general.
But FFT-style mixings are *guaranteed* to propagate that 1-bit change
to all intermediate results. And, although this is not exactly what
we want, each of these select a new entry in their own random table,
which, when mixed one more time, *does* give us what we want. This is
a scalable and measurable result, from a construction using 2 small
component types in a simple structure very similar to that used in the
well known FFT. What is difficult about this?

>and I'd be
>interested to know what disadvantages you see.

Before seeing your proposal, I would first question some part of:
speed, quality of f(), number of rounds, and quality of diffusion.

>
>http://www.hedonism.demon.co.uk/paul/mercy

This certainly is a nice presentation. It is easy to get around in,
and probably much easier to maintain than my web stuff. I like the
generally sparse nature of the articles, which seem qualitatively
different than paper articles translated to html. I do think the
variety of components in this design make it hard to understand at
first shot, however, and I think the articles could also better help
the reader to reason about the ciphering process.

After seeing your proposal, the first thing I would like to know
concerns actual measurements: What kind of speeds are you seeing (on
what platform), and have you done any diffusion testing? In what
sense and to what extent does your construction emulate a large Simple
Substitution?

Also it would seem that Mixing ciphers do have the property described
in the justification, have been described many times right here, and
recommended to you particularly, so one might well wonder why they are
not cited as an alternative.

Also it looks like the main nonlinearity in the system is the single 8
to 32 bit "T" table, and that makes me nervous. This is 1KB of total
state, and the differential input to and output from that table are
directly visible in the last round. That makes me nervous.

>> Even here on sci.crypt I have often seen the opinion expressed that a
>> large block cipher simply could not provide the necessary mixing or
>> diffusion. The presentation of FFT-style mixing with log n levels is
>> at least a plausible counterexample, but it has not appeared in the
>> texts.
>
>One design I've played with uses an F function with data-dependent
>indexing into the right half; this isn't used in the cipher at the
>moment, but does appear in the current revision of the key schedule.
>I suspect that with work such a construction could provide diffusion
>as rapidly as the rotates in RC5; however, it pretty much rules out a

>bijective F-function so far as I can tell. You even could shuffle the
>words in the left half using randomising information taken from the
>right half, though I suspect that data-dependent indexing provides the
>same advantages.

Yes. I am aware of other mixing approaches which avalanche. But I suspect that we can sort them into 4 piles:

* n log n effort "full" balanced mixing,

* more effort,

* less mixing,

* n effort "constant depth" mixing.

We ought to be able to more or less measure this stuff, in scaled-down implementations. But if we can't scale it down, and can't measure it, a complex design has very serious believability problems.

Terry Ritter ritter@io.com http://www.io.com/~ritter/
Crypto Glossary 1998-08-27: http://www.io.com/~ritter/GLOSSARY.HTM

Subject: Re: Thought question: why is encrypted output routinely the same size as the input?
Date: Fri, 16 Oct 1998 18:47:07 +0100
From: Mok-Kong Shen <mok-kong.shen@stud.uni-muenchen.de>
Message-ID: <3627869B.DA0DA918@stud.uni-muenchen.de>
References: <36224706.5167449@news.io.com>
Newsgroups: sci.crypt
Lines: 59

Terry Ritter wrote:

> My guess is that the avoidance of large blocks is first related to the
> use of Feistel constructions for block ciphers. Feistel constructions
> need a half-block-wide f() function, which ideally is 1:1 (a
> permutation) as in DES. It is just difficult to construct ciphers
> with large blocks using "normal" technology.

I suppose to say more plainly the cause is that at the time of invention of the Feistel cipher fast software encryption was not yet available. So one concentrated on hardware and hardware was also expensive then. This led to very small block size and Feistel has apparently succeeded to make the best out of that by operating on the bit level in an ingenious way. (If one uses a fairly large block size, operating on the bit level would be too costly or complicated and one would choose larger units for operations.)

>
> Even here on sci.crypt I have often seen the opinion expressed that a
> large block cipher simply could not provide the necessary mixing or
> diffusion. The presentation of FFT-style mixing with log n levels is
> at least a plausible counterexample, but it has not appeared in the
> texts.

In Schneier's book there is only a reference to FFT-Hash. Evidently you mean something different. Could you give the relevant references?

> Another problem is a general lack of knowledge of the advantages of
> large blocks. When you have a hammer, you tend to see all your
> problems as nails, and here people have a particular vision of a
> DES-like block cipher. Changing that view requires changing many
> other things about the way ciphers operate and which had been thought
> fully resolved. Some of the advantages of large blocks are best
> realized by system re-design, as opposed to simply dropping a new
> implementation into the DES-box slot.

But even some classical transpositions can be considered to operate on block sizes much larger than 64 bits. So one seems to have been somehow dazzeled by the intensity of light emitted from DES. Further the distinction between stream and block cipher is not absolute but a matter of terminological convenience, if I don't err.

> I suppose tradeoffs are inherently more engineering than general
> academics, but we don't have a good engineering text on crypto, and
> the academic texts have uniformly failed to explore the possibilities
> of new structures and new designs. Perhaps after AES we will see
> texts with wider horizons.

If there are no new structures and new designs invented or better deployment of old stuffs discovered, you can't blame the authors of the texts for failing to treat them. The volume of publications in cryptology is yet small compared to most fields of computer science. There are only a couple of journals. Your opinion on the influence of AES in the said direction appears to be an over-estimation for me.

M. K. Shen

Subject: Re: Thought question: why is encrypted output routinely the same size as the input?
Date: Sat, 17 Oct 1998 04:07:29 GMT
From: ritter@io.com (Terry Ritter)
Message-ID: <362817e5.3263247@news.io.com>
References: <3627869B.DA0DA918@stud.uni-muenchen.de>
Newsgroups: sci.crypt
Lines: 64

On Fri, 16 Oct 1998 18:47:07 +0100, in
<3627869B.DA0DA918@stud.uni-muenchen.de>, in sci.crypt Mok-Kong Shen
<mok-kong.shen@stud.uni-muenchen.de> wrote:

>[...]
>> Even here on sci.crypt I have often seen the opinion expressed that a
>> large block cipher simply could not provide the necessary mixing or
>> diffusion. The presentation of FFT-style mixing with log n levels is
>> at least a plausible counterexample, but it has not appeared in the
>> texts.
>
>In Schneier's book there is only a reference to FFT-Hash. Evidently
>you mean something different. Could you give the relevant references?

http://www.io.com/~ritter/ARTS/NONLBBM.HTM#1998092201
http://www.io.com/~ritter/#MixTech

>[...]
>Further the distinction between stream and block cipher is not
>absolute but a matter of terminological convenience, if I don't err.

Most crypto scientists see a block cipher as an emulated huge Simple Substitution. So if a "stream cipher" had that characteristic, it would not *be* a stream cipher, no matter what the convenience. It is certainly tempting to use this definition, because keyed or randomized Simple Substitution has a diffusion signature which a stream cipher cannot have. This would be a practical and conclusive experimental difference.

Nevertheless, I use a slightly different definition: I see a block cipher as one which requires the accumulation of data before ciphering can occur. This intuitive definition is consistent with the term "block," but it also means that transposition is a form of block cipher, and transposition does not have the diffusion signature.

Either definition allows us to distinguish a stream cipher mainly as "not a block cipher," in particular, a cipher which does *not* require the accumulation of data before ciphering can occur. We can also see that stream ciphers only diffuse changes to later cipherings.

Then I see block cipher "chaining modes" (like CBC) as stream meta-ciphers which use a block cipher as a component. This provides a welcome unity between the concepts used in classical stream cipher designs and current block cipher usage.

>[...]
>If there are no new structures and new designs invented or better
>deployment of old stuffs discovered, you can't blame the authors of
>the texts for failing to treat them.

This is one sentiment to which I am particularly sensitive. I have been inventing fundamentally new cryptography and writing about it (sometimes in peer-reviewed ink-on-paper articles) for about a decade. But I have yet to see any of this stuff appear in a text. So you tell me: Should I *not* blame the authors?

Terry Ritter ritter@io.com http://www.io.com/~ritter/
Crypto Glossary 1998-08-27: http://www.io.com/~ritter/GLOSSARY.HTM

Subject: Re: Thought question: why is encrypted output routinely the same size as the input?
Date: Sat, 17 Oct 1998 22:53:42 -0600
From: jgfunj@EnqvbSerrGrknf.pbz (W T Shaw)
Message-ID: <jgfunj-1710982253430001@dialup175.itexas.net>
References: <362817e5.3263247@news.io.com>
Newsgroups: sci.crypt
Lines: 36

In article <362817e5.3263247@news.io.com>, ritter@io.com (Terry Ritter) wrote:
>
> Most crypto scientists see a block cipher as an emulated huge Simple
> Substitution. So if a "stream cipher" had that characteristic, it
> would not *be* a stream cipher, no matter what the convenience. It is
> certainly tempting to use this definition, because keyed or randomized
> Simple Substitution has a diffusion signature which a stream cipher
> cannot have. This would be a practical and conclusive experimental

> difference.

Severing a stream of data into blocks for the convenience of handling should be included. In particular where a restarting of some process with each such chunk is involved.
>
> Nevertheless, I use a slightly different definition: I see a block
> cipher as one which requires the accumulation of data before ciphering
> can occur. This intuitive definition is consistent with the term
> "block," but it also means that transposition is a form of block
> cipher, and transposition does not have the diffusion signature.

Neither would the keyed substitutions I use, where no overlaping transposition is involved. The block construction, or text segmentation, is more of a convenience for the key to keep them from beeing too large to be easily implemented.
>
> Either definition allows us to distinguish a stream cipher mainly as
> "not a block cipher," in particular, a cipher which does *not* require
> the accumulation of data before ciphering can occur. We can also see
> that stream ciphers only diffuse changes to later cipherings.

I accept that as a great definition.
--

Insanity means doing the same thing over and over again and expecting different results...like CDA2.

Decrypt with ROT13 to get correct email address.

Subject: Re: Thought question: why is encrypted output routinely the same size as the input?
Date: Mon, 19 Oct 1998 16:20:33 +0100
From: Mok-Kong Shen <mok-kong.shen@stud.uni-muenchen.de>
Message-ID: <362B58C1.45F3D896@stud.uni-muenchen.de>
References: <362817e5.3263247@news.io.com>
Newsgroups: sci.crypt
Lines: 70

Terry Ritter wrote:
>
> >> The presentation of FFT-style mixing with log n levels is
> >> at least a plausible counterexample, but it has not appeared in the
> >> texts.

> http://www.io.com/~ritter/ARTS/NONLBBM.HTM#1998092201
> http://www.io.com/~ritter/#MixTech

I have a couple of tiny questions: Where is the link (in meaning) between orthogonal Latin squares and FFT (which has an established definition in numerical mathematics)? Where does the 'log n levels' come into the encryption process?

> Nevertheless, I use a slightly different definition: I see a block
> cipher as one which requires the accumulation of data before ciphering
> can occur. This intuitive definition is consistent with the term
> "block," but it also means that transposition is a form of block
> cipher, and transposition does not have the diffusion signature.

I prefer simply to think that a block cipher operate on units of blocks (bunches of characters or bits (which are the fundamental units)). This (larger) unit, the block, must of course be complete before being operated on. So your definition is inline with mine. A transposition is certainly a block cipher then (independent of whether diffusion is achieved). If we XOR two character streams, it would according to my definition be a stream cipher if the fundamental unit is character and a block cipher if the fundamental unit is bit. The choice of the fundamental unit and the larger unit (block) is essentially arbitrary, depending on the standpoint one prefers. (One can also choose to consider a word of 32 bits or a group of 64 bits to be a fundamental unit.) This is what I meant by there being no absolute distinction between stream and block cipher.

> Then I see block cipher "chaining modes" (like CBC) as stream
> meta-ciphers which use a block cipher as a component. This provides a
> welcome unity between the concepts used in classical stream cipher
> designs and current block cipher usage.

Taking the block as the fundamental unit CBC is a stream cipher.
So we agree.

> >If there are no new structures and new designs invented or better
> >deployment of old stuffs discovered, you can't blame the authors of
> >the texts for failing to treat them.
>
> This is one sentiment to which I am particularly sensitive. I have
> been inventing fundamentally new cryptography and writing about it
> (sometimes in peer-reviewed ink-on-paper articles) for about a decade.
> But I have yet to see any of this stuff appear in a text. So you tell
> me: Should I *not* blame the authors?

Apology for having been provocative. If I remember correctly, Bruce Schneier wrote recently in a post somewhere that cryptology is still a black art. Given this fact and remembering the history of development of natural sciences, one should nonetheless have some good understanding of the phenomenon that hurts you. Anyway, Web publication has helped to disseminate certain ideas that would otherwise have been totally suppressed (particularly to the satisfaction of certain crypto bureaucrats). So the time is certainly much better than that of our far ancestors. Discussion groups like sci.crypt and some mailing lists have further helped quite a bit. I think there is a need though of a site(s) that gives comprehensive links to Web publications (not software packages, nor simply reprints from journals) on cryptology so that one can conveniently find them, since the common search machines are ineffective, I believe.

M. K. Shen

Subject: Re: Thought question: why is encrypted output routinely the same size as the input?
Date: Mon, 19 Oct 1998 11:50:52 -0600
From: jgfunj@EnqvbSerrGrknf.pbz (W T Shaw)
Message-ID: <jgfunj-1910981151080001@207.101.116.84>
References: <362B58C1.45F3D896@stud.uni-muenchen.de>
Newsgroups: sci.crypt
Lines: 106

In article <362B58C1.45F3D896@stud.uni-muenchen.de>, Mok-Kong Shen <mok-kong.shen@stud.uni-muenchen.de> wrote:
>
> I prefer simply to think that a block cipher operate on units of blocks
> (bunches of characters or bits (which are the fundamental units)).

I wrote some block ciphers that have nothing to do with bits some time ago; it is based on trits. One version will retain capitalization, and handle word spacing, Trinity 27. Punctuation is substituted with rare characters, X, J, and Q.

I'll use the above couple of sentences as the source material. Hashing the text gets these keys, two substitution and one transposition.

Sub1: k/tpxevqo fbnrcmdls hguwyiazj
Sub2: azdmougin fvpcq/jwh bkxstrley
Trans: jneaiqkxl ucbvfdroy tmzwspg/h

27 characters targeted in the set means each one is coded to 3 trits; the values are assigned according to placement in the first substitution key. Groups of 9 characters mean 27 trits per block. Each block is rearranged according to the transposition key. The 9 groups of 3 trits in each block are converted to strings using the second substitution key.

Here is the plaintext after automatic conversion of numbers to words, and preformatting the bulk of the text:

I/wrote/s ome/block /ciphers/ that/have /nothing/ to/do/wit h/bits/so
me/time/a go/it/is/ based/on/ tritsX/On e/version /will/ret ain/capit
alization j/and/han dle/word/ spacingj/ Trinity/t wo/sevenX /Punctuat
ion/is/su bstituted /with/rar e/charact ersj/Xj/J j/and/QX/

Here are the encrypted groups:

z Zyjlcczzy cjzdywfqj vmonvsstb ljggbdldz jmmdzhmyl dynzflwgf ibgtdtxmr
gekvqwaby isukflymj euu/bba/j jywnzBbVc iija/tlpl r/infbqvf svirhyfja
/dhyjsykl gnjlgmehf czuphmstb oeojgkeys Jeinffiks nkuk/lzmI gQinaaejl
meyruebss wsppaypay lmvnfasev angotqhta hjeutFndH gnmllcVPc

Here is the recovered text:

I wrote some block ciphers that have nothing to do with bits some time ago
it is based on tritsX One version will retain capitalizationj and handle
word spacingj Trinity two sevenX Punctuation is substituted with rare
charactersj Xj Jj and QX

To keep it simple, I made some compromises, but there is narry a bit it
the whole thing.

Adding a trit to each character length means having a character set of 81, upper, lower, and symbol case. The transposition key for Trinity 36 is a little longer since there are 36 trits in each block, and I will not go into all the details, but this scaled-up version of Trinity 27 does work.

Using the above text to generate the keys, they are:

Sub1: aijpsmbxu qyztkcfng odveh/lwr
Sub2: hrkymqias wnb/ptlcf oduegzjvx
Trans: g2sl985vw hxnt6iyaq f/7pduoeb 1j3mk4zcr

The preformed text is:

Adding!a! trit!to!e ach!chara cter!leng th!means! having!a! character
!set!of!8 1,!upper, !lower,!a nd!symbol !case.!Th e!transpo sition!ke
y!for!Tri nity!36!i s!a!littl e!longer! since!the re!are!36 !trits!in
!each!blo ck,!and!I !will!not !go!into! all!the!d etails,!b ut!this!s
caled-up! version!o f!Trinity !27!does! work.fill

The encrypted groups are:

```
+yW,jK8K; orsKHHq2( WIiO+Ko9A uNe2,kscV edoclkhT~ +qw,v,8by ,njR,hiT7
dU*rIkqKc rw0XlsyAD 5b6_+k8h- +glIrkiU+ dk@,wo;$C YBo88aht5 OH/2rr;Ty
d3id4vHi9 4DfRRd,Oy Owyr4ijt8 +(i)0shf; o+jnNDyTl osqXxk9g1 N,+Bw,9rR
oz0hWohs3 MRs*$RoWM eu9r!OyR- Ht9_9ko,Q N+sK+kok* EbfRoh9wR G/siBkyoN
J7h8*uhtw y:frk,hk3 MMe;8ahg5 o(8f91oqs e,l8D,sx=
```

The recovered text is:

Adding a trit to each character length means having a character set of 81, upper, lower, and symbol case. The transposition key for Trinity 36 is a little longer since there are 36 trits in each block, and I will not go into all the details, but this scaled-up version of Trinity 27 does work.fill

---

Bits are particularily unfriendly to the realization of this version of the Trinity cipher. Now, this was not meant to be the strongest thing around, but it is very useful for something fairly short., much better than most old classics. I classify these types of algorithms as *neoclassical*, utilizing calculations that would are best done on a computer but could be clearly demonstrated, if laboriously so, with physical models.

Something simple like this could be made extended in a trit version takeoff of a common round based ciphers, even something modified from a Feistel construction, but it was my sole purpose to make the good point that many do not cheerfully accept, that a bit is just another unit of logic, nothing special or superior about it at all. What makes the Trinity Ciphers block ciphers has nothing to do with the bit, and the concept in them is somewhat scalable as a bonus.
--

Insanity means doing the same thing over and over again and expecting different results...like CDA2.

Decrypt with ROT13 to get correct email address.

Subject: Re: Thought question: why is encrypted output routinely the same size as the input?
Date: Mon, 19 Oct 1998 19:02:22 +0100
From: Mok-Kong Shen <mok-kong.shen@stud.uni-muenchen.de>
Message-ID: <362B7EAE.181E7245@stud.uni-muenchen.de>
References: <jgfunj-1910981151080001@207.101.116.84>
Newsgroups: sci.crypt
Lines: 19

W T Shaw wrote:
>
> In article <362B58C1.45F3D896@stud.uni-muenchen.de>, Mok-Kong Shen
> <mok-kong.shen@stud.uni-muenchen.de> wrote:
> >
> > I prefer simply to think that a block cipher operate on units of blocks
> > (bunches of characters or bits (which are the fundamental units)).
>
> I wrote some block ciphers that have nothing to do with bits some time
> ago; it is based on trits. One version will retain capitalization, and
> handle word spacing, Trinity 27. Punctuation is substituted with rare
> characters, X, J, and Q.

Just an addendum to my sentence: I mean both character and bit, depending on one's preference, can serve as the fundamental unit in discourse. Generalizing, also ideographs (the Chinese characters) or similar entities could be considered units.

M. K. Shen

Subject: Re: Thought question: why is encrypted output routinely the same size as the input?
Date: Mon, 19 Oct 1998 17:54:28 GMT
From: ritter@io.com (Terry Ritter)
Message-ID: <362b7c6f.5322932@news.io.com>
References: <362B58C1.45F3D896@stud.uni-muenchen.de>
Newsgroups: sci.crypt
Lines: 220

On Mon, 19 Oct 1998 16:20:33 +0100, in
<362B58C1.45F3D896@stud.uni-muenchen.de>, in sci.crypt Mok-Kong Shen
<mok-kong.shen@stud.uni-muenchen.de> wrote:

>Terry Ritter wrote:
>>
>> >> The presentation of FFT-style mixing with log n levels is
>> >> at least a plausible counterexample, but it has not appeared in the
>> >> texts.
>
>> http://www.io.com/~ritter/ARTS/NONLBBM.HTM#1998092201
>> http://www.io.com/~ritter/#MixTech

>
>I have a couple of tiny questions: Where is the link (in meaning)
>between orthogonal Latin squares and FFT (which has an established
>definition in numerical mathematics)? Where does the 'log n levels'
>come into the encryption process?

OK, I'm going to explain this in different words, but first here is the section from the recent OLS article, and if that does not make this clear on its own, please try to help me understand why:

"A TYPICAL MIXING CIPHER"

"In figure 1 we have a typical Mixing cipher in schematic form, with 3 "fencing" layers of 8 invertible substitutions each, and two full "FFT-style" mixings between them. If these are 8-bit substitutions, we have a 64-bit block cipher. Each substitution (and now each mixing operation also) is individually keyed.

"The vertical dotted lines represent typically 8-bit-wide data paths, and the data flow is from top to bottom. Each S is a substitution table, and *--* represents the "butterfly" action of a single BBM. For 8 elements we have log2(8) = 3 mixing FFT sublayers (Mix0, Mix1, Mix2) of 8/2 = 4 BBM operations each. All BBM's in the same sublayer are independent and can occur simultaneously in parallel hardware. The wider the block, the more BBM's needed, which also means that more computation can occur simultaneously.

```
|   A 64-Bit Mixing Cipher    Fig. 1
|
|    :   :   :   :   :   :   :   :   &LT;- Input Block
|    S   S   S   S   S   S   S   S   &LT;- Fencing
|    :   :   :   :   :   :   :   :
|    *---*   *---*   *---*   *---*    Mix0 0..0, 0..3
|    :   :   :   :   :   :   :   :
|    *-------*   :   *-------*   :    Mix1 0..0, 0..1
|    :   *-------*   :   *-------*    Mix1 1..1, 0..1
|    :   :   :   :   :   :   :   :
|    *---------------*   :   :   :    Mix2 0..0, 0..0
|    :   *---------------*   :   :    Mix2 1..1, 0..0
|    :   :   *---------------*   :    Mix2 2..2, 0..0
|    :   :   :   *---------------*    Mix2 3..3, 0..0
|    :   :   :   :   :   :   :   :
|    S   S   S   S   S   S   S   S   &LT;- Fencing
|    :   :   :   :   :   :   :   :
|    *---------------*   :   :   :
|    :   *---------------*   :   :
|    :   :   *---------------*   :
|    :   :   :   *---------------*
|    :   :   :   :   :   :   :   :
|    *-------*   :   *-------*   :
|    :   *-------*   :   *-------*
|    :   :   :   :   :   :   :   :
|    *---*   *---*   *---*   *---*
|    :   :   :   :   :   :   :   :
|    S   S   S   S   S   S   S   S   &LT;- Fencing
|    :   :   :   :   :   :   :   :   &LT;- Output Block
```

"Each of the mixing BBM's uses an orthogonal pair of Latin squares. If these are nonlinear, we must of course process the FFT-style "sublayers" in reverse order when deciphering. So if we use the opposite orders in the top and bottom mixing sections, we can use the exact same structure for both enciphering and deciphering; only the table contents then need be changed."

OK, that was the old section, now for some more:

Take a look at any of the diagrams, including the nicer .GIF ones on my pages, which start at the top left of my main page. (Click on that and you get to the bigger one some ways down. Click on that and you get to the html article on this mixing and an even bigger diagram.)

These diagrams show data flowing from top to bottom on lines which transport one element each (that would be a single value, possibly represented in binary on 8 copper wires). Now, see the horizontal lines? They each connect two lines or elements which are to be mixed together.

Each of these mixings is one BBM or pair of orthogonal Latin squares, and takes two elements, mixes them, and returns two mixed values. The mixed results then replace the original values in the same positions just like the "butterfly" operations used in some FFT's.

So we can mix each element with another, and then each pair with another pair and so on until every element is mixed with every other element. This occurs in log2(n) "sub-levels," because each BBM mixing is "dyadic" in the sense of taking 2 inputs, and so each mixing sublevel has the mixed results from twice as many elements as the sublevel before it. So we can obviously produce a pair of elements which each contain "an equal contribution" from n original input

values in 2**m steps, where m = log2(n).

And if we do this across the full width of the block, we get a full block of mixed elements in which each of the original block elements is represented "equally" in each element of the result. This block is decipherable to the same extent that each of the mixings is reversible. And this is guaranteed by the nature of the mixings.

The pattern of these mixings is exactly like some implementations of the FFT, which is thus "FFT-style."

>> Nevertheless, I use a slightly different definition: I see a block
>> cipher as one which requires the accumulation of data before ciphering
>> can occur. This intuitive definition is consistent with the term
>> "block," but it also means that transposition is a form of block
>> cipher, and transposition does not have the diffusion signature.
>
>I prefer simply to think that a block cipher operate on units of blocks
>(bunches of characters or bits (which are the fundamental units)

Bits are the fundamental units of *information*, but not necessarily the fundamental units of *operation*, which could be any representation, such as "trits" or whatever.

>). This
>(larger) unit, the block, must of course be complete before being
>operated on. So your definition is inline with mine. A transposition
>is certainly a block cipher then (independent of whether diffusion
>is achieved). If we XOR two character streams, it would according to
>my definition be a stream cipher if the fundamental unit is character
>and a block cipher if the fundamental unit is bit. The choice of the
>fundamental unit and the larger unit (block) is essentially arbitrary,
>depending on the standpoint one prefers. (One can also choose to
>consider a word of 32 bits or a group of 64 bits to be a fundamental
>unit.) This is what I meant by there being no absolute distinction
>between stream and block cipher.

I still think this would generally be considered a wrong interpretation of streaming. When true streaming occurs it is boundless. Now, we might have a system which for its own reasons inserts boundaries and so streams in "chunks" (this is *not* a block cipher), but the streaming itself is boundless.

If the distinction between "block" and "stream" is going to have any meaning, streaming across a 64-bit block is still streaming: From the point of view of the streaming operation, the 64-bit block of data did not need to all be available at one time. A "true" block cipher must have a full block at once for operations to proceed.

I note that the inability to make this distinction would essentially invalidate the expanded form of the block definition, and so return us to the emulated Simple Substitution definition (which is already the dominant understanding). The problem with *that* is that it says "block" for a distinction which is *not* based on the "block" but instead based on *a* *particular* *type* of block transformation. And *that* means that we need a name for "block" transformations of *other* types, and *that* will be very, very confusing.

>> Then I see block cipher "chaining modes" (like CBC) as stream
>> meta-ciphers which use a block cipher as a component. This provides a
>> welcome unity between the concepts used in classical stream cipher
>> designs and current block cipher usage.
>
>Taking the block as the fundamental unit CBC is a stream cipher.
>So we agree.

Great!

>I think there is a need though of a
>site(s) that gives comprehensive links to Web publications (not
>software packages, nor simply reprints from journals) on cryptology
>so that one can conveniently find them,

Well, possibly, but that site would be very difficult to maintain. Ideally we might think that for some reason every author would want to make sure that every one their works was there, but that won't happen.

On the other hand, every author nowadays should have some sort of home page which links to their particular articles. Each author should have the responsibility of making their particular articles available. So then the problem becomes finding those home pages. I have links to a few lists of crypto people's home pages in my crypto links.

I think a better idea would be to have short reviews of each article, organized by topic. I have been doing some of this for ink-on-paper articles in the "Literature Surveys" section on my pages:

http://www.io.com/~ritter/#LiteratureSurveys

In this way one can find the articles *on a particular topic* in one place, and from the quoted text make a meaningful selection of those

which might be worth pursuing, whether on-line or in a library. Doing this for every topic in cryptography would, however, be a very big and generally thankless job.

>since the common search
>machines are ineffective, I believe.

Yeah, I don't know what happened. It seemed like they were far more useful even just a year ago. Now it seems like one can hardly find anything. Odd.

Terry Ritter ritter@io.com http://www.io.com/~ritter/
Crypto Glossary http://www.io.com/~ritter/GLOSSARY.HTM

Subject: Re: Thought question: why is encrypted output routinely the same size as the input?
Date: 20 Oct 1998 04:26:57 GMT
From: jsavard@freenet.edmonton.ab.ca ()
Message-ID: <70h3eh$ek6$2@news.sas.ab.ca>
References: <362b7c6f.5322932@news.io.com>
Newsgroups: sci.crypt
Lines: 21

Terry Ritter (ritter@io.com) wrote:
: Yeah, I don't know what happened. It seemed like they were far more
: useful even just a year ago. Now it seems like one can hardly find
: anything. Odd.

I have noticed things have gotten worse since I've started using the Internet - but since I started recently, and even more recently had a Web page, I think I know why.

For one thing, many web pages are not permanent, and move to locations. For another, the number of web sites is growing at a rate the search engines are finding it hard to keep up with.

As a result, they're indexing a smaller proportion of existing Web sites, and a larger number of out-of-date links. Plus, many of the services are being funded, or advertisers are trying them, on a speculative basis. If banner ads on AltaVista or Yahoo! don't bring the results advertisers expect, eventually the funds for the elaborate computer installations needed to provide these free services will dry up.

John Savard

Subject: Re: Thought question: why is encrypted output routinely the same size as the input?
Date: Tue, 20 Oct 1998 11:00:06 +0100
From: Mok-Kong Shen <mok-kong.shen@stud.uni-muenchen.de>
Message-ID: <362C5F26.D347BAB3@stud.uni-muenchen.de>
References: <362b7c6f.5322932@news.io.com>
Newsgroups: sci.crypt
Lines: 120

Terry Ritter wrote:
>
> <mok-kong.shen@stud.uni-muenchen.de> wrote:
.....

> Bits are the fundamental units of *information*, but not necessarily
> the fundamental units of *operation*, which could be any
> representation, such as "trits" or whatever.

We indeed agree. A unit of operation can consist of several (arbitrarily defined) fundamental units of information. Such a unit I call a block.

> > This
> >(larger) unit, the block, must of course be complete before being
> >operated on. So your definition is inline with mine. A transposition
> >is certainly a block cipher then (independent of whether diffusion
> >is achieved). If we XOR two character streams, it would according to
> >my definition be a stream cipher if the fundamental unit is character
> >and a block cipher if the fundamental unit is bit. The choice of the
> >fundamental unit and the larger unit (block) is essentially arbitrary,
> >depending on the standpoint one prefers. (One can also choose to
> >consider a word of 32 bits or a group of 64 bits to be a fundamental
> >unit.) This is what I meant by there being no absolute distinction
> >between stream and block cipher.
>
> I still think this would generally be considered a wrong
> interpretation of streaming. When true streaming occurs it is
> boundless. Now, we might have a system which for its own reasons
> inserts boundaries and so streams in "chunks" (this is *not* a block
> cipher), but the streaming itself is boundless.

We seem to agree here too. My sentences nowhere contradict the thesis that a stream is boundless. A stream in my definition is simply an unspecified number of (chosen) fundamental units of information. It has no boundaries or 'chunks'. If we impose a (again arbitrary)

block structure on it (see my definition) than there are boundaries
of the blocks (delimiting those fundamental units belonging to each
individual block) but the stream itself has no such boundaries.
A practical stream of couse has on the other hand a start and an end
(message begin and message end).

>
> If the distinction between "block" and "stream" is going to have any
> meaning, streaming across a 64-bit block is still streaming: From the
> point of view of the streaming operation, the 64-bit block of data did
> not need to all be available at one time. A "true" block cipher must
> have a full block at once for operations to proceed.

I don't see disagreement between us. As I tried to explain, the same
information flow can be both regarded as stream or blocks. If you
are doing a streaming operation, you operate on each (of your defined)
fundamental unit at a time. If that unit is the bit then all you
need is one bit at one time. If the supplier gives you more bits
simutaneously, that's fine but you don't need that extra service.
A block cipher operates on one block at a time and by definition
must have all the fundamental units belonging to it available before
the operation can work. On this last point we evidently fully agree.

> I note that the inability to make this distinction would essentially
> invalidate the expanded form of the block definition, and so return us
> to the emulated Simple Substitution definition (which is already the
> dominant understanding). The problem with *that* is that it says
> "block" for a distinction which is *not* based on the "block" but
> instead based on *a* *particular* *type* of block transformation. And
> *that* means that we need a name for "block" transformations of
> *other* types, and *that* will be very, very confusing.

Having said above I would like to add only that the grouping of
fundamential units into blocks is 'syntatic'. The opearation done on
a block is 'semantic' and is entirely dependent on the cipher in
question. It can consist of more or less complex (sub-)operations
on the constituent fundamental units. To look at the mapping from
the (pre-)operated block to the block after the enciphering process
simply as a substitution is a legitimate but extremely oversimplified
view of the matter, because it renders all the (sub-)operations
carried out in the processing invisible.

> >I think there is a need though of a
> >site(s) that gives comprehensive links to Web publications (not
> >software packages, nor simply reprints from journals) on cryptology
> >so that one can conveniently find them,
>
> Well, possibly, but that site would be very difficult to maintain.
> Ideally we might think that for some reason every author would want to
> make sure that every one their works was there, but that won't happen.

Well, I tend to think that in the actual situation something of that
sort is still better than nothing. Of course the site cannot link
to just anything. Some more or less liberal policies as those commonly
practiced by some mailing lists suffice to avoid certain undesirable
problems, I guess. Unfortunately for technical reasons I am unable
to offer that service on my site, otherwise I would have started such
a project. Maintenace wouldn't be a big issue. If some links become
unavailable with time one can just forget about them until someone
complains.

>
> On the other hand, every author nowadays should have some sort of home
> page which links to their particular articles. Each author should
> have the responsibility of making their particular articles available.
> So then the problem becomes finding those home pages. I have links to
> a few lists of crypto people's home pages in my crypto links.
>
> I think a better idea would be to have short reviews of each article,
> organized by topic. I have been doing some of this for ink-on-paper
> articles in the "Literature Surveys" section on my pages:
>
>   http://www.io.com/~ritter/#LiteratureSurveys
>
> In this way one can find the articles *on a particular topic* in one
> place, and from the quoted text make a meaningful selection of those
> which might be worth pursuing, whether on-line or in a library. Doing
> this for every topic in cryptography would, however, be a very big and
> generally thankless job.

Your recommendations are cretainly very reasonable. I see you have
done a good job.

M. K. Shen

Subject: Re: Thought question: why is encrypted output routinely the same size as the
input?
Date: Thu, 08 Oct 1998 11:31:10 +0100
From: Mok-Kong Shen <mok-kong.shen@stud.uni-muenchen.de>
Message-ID: <361C946E.9BC8501E@stud.uni-muenchen.de>
References: <6v5mcq$cj1$1@nnrp1.dejanews.com>
   <3613D2D3.9F597176@stud.uni-muenchen.de>

Newsgroups: sci.crypt
Lines: 47

dianelos@tecapro.com wrote:
>
> In article <3613D2D3.9F597176@stud.uni-muenchen.de>,
> Mok-Kong Shen <mok-kong.shen@stud.uni-muenchen.de> wrote:
> >[...]
> > I am not yet quite clear of how your scheme compare with the
> > following:
> >
> > use a key (known to the partner) and PRNG to generate R
> > but do not transmit R.
> > compute and transmit C = Encrypt(T xor R)
> >
> Well, the two methods are clearly different. Your method does use
> a larger key but does not inject noise and therefore does not
> increase the size of the ciphertext. In fact, if the PRNG is

I see that I haven't yet properly understood your idea. Let me
therefore reproduce your previous post:

> Noise can easily be used with any cipher. Here is a possibility:
> Use noise to initialize a RNG and XOR its output with plaintext
> blocks before encrypting. Repeat this process periodically. If
> your RNG has an internal state of 128 bits (16 bytes) and you
> re-initialize it with noise every 512 bytes of plaintext then you
> increase the size of the ciphertext only 3.1%

> produce and transmit noise R
> compute and transmit C = Encrypt(T xor R)

You use noise often times as seed to a PRNG to obtain its output R
and XOR the plaintext T with R. (The phrase 'noise R' above seems to
be 'output R of PRNG using some noise as seeds'.) You encrypt (T XOR R)
to C and send both R and C. So you double the volume of transmission.
My point is that R is available to the analyst and this is
undesirable.

So I think one of your sentences above probably should read:

 use some noise N via PRNG to generate R and transmit N
 compute and transmit C = Encrypt(T XOR R)

But then I think one should transmit not N but U = Encrypt(N).

Anyway I am much confused and should very much appreciate your
explanation.

M. K. Shen

Subject: Re: Thought question: why is encrypted output routinely the same size as the
input?
Date: Thu, 08 Oct 1998 19:24:31 GMT
From: dianelos@tecapro.com
Message-ID: <6vj3hfjii1@nnrp1.dejanews.com>
References: <361C946E.9BC8501E@stud.uni-muenchen.de>
Newsgroups: sci.crypt
Lines: 78

In article <361C946E.9BC8501E@stud.uni-muenchen.de>,
  Mok-Kong Shen <mok-kong.shen@stud.uni-muenchen.de> wrote:
> dianelos@tecapro.com wrote:
> > produce and transmit noise R
> > compute and transmit C = Encrypt(T xor R)
>
> You use noise often times as seed to a PRNG to obtain its output R
> and XOR the plaintext T with R. (The phrase 'noise R' above seems to
> be 'output R of PRNG using some noise as seeds'.) You encrypt (T XOR R)
> to C and send both R and C. So you double the volume of transmission.
> My point is that R is available to the analyst and this is
> undesirable.

  Yes and no. Suppose a new random R is produced with each encryption.
  Then this simple method at least stops all *chosen* plaintext
  attacks. R is known after the encryption but an attacker can
  only choose T before the encryption. This method does not stop
  chosen ciphertext attacks, where the attacker can specify both R
  and C.

> So I think one of your sentences above probably should read:
>
> use some noise N via PRNG to generate R and transmit N
> compute and transmit C = Encrypt(T XOR R)
>
> But then I think one should transmit not N but U = Encrypt(N).
>
> Anyway I am much confused and should very much appreciate your
> explanation.

  Here is what I want: A method that uses ciphers as a primitive
  and stops all known plaintext or ciphertext only attacks. I don't
  see why such a method cannot exist; if it turns out that such a

method does exist and is practical it would be extremely important
for many security systems where speed is not critical.

Terry Ritter proposes to include a random field in the text block.
This does have advantages but it does not stop ciphertext only attacks
which are based on partial knowledge of the plaintext.

The method I proposed earlier in this thread (with the double
encryption excised - as Terry said using double encryption
cannot be functionally different from using single encryption) is
this:

```
        For each plaintext T generate random R and compute:
        C1 = E( k1, R )
        C2 = E( k2, R xor T )
```

The idea is that the second encryption uses plaintexts that are
in principle unknown to the attacker.

Observe that it is possible to cancel out the random variable R:

```
        R = D( k1, C1 ) =>

        C2 = E( k2, D( k1, C1 ) xor T )
```

This last expression can be rewritten thus:

```
        C2 = F( k1,k2,C1, T )
```

where F is a new cipher with a key that is conformed by the
secret but constant k1,k2, and the known but variable C1.
Therefore this method is equivalent to using a cipher with
variable keys, which I understand does stop known plaintext or
ciphertext only analysis.

So this method may work for most ciphers E but I don't know how
to prove it.

--
http://www.tecapro.com
email: dianelos@tecapro.com

----------== Posted via Deja News, The Discussion Network ==----------
http://www.dejanews.com/ Search, Read, Discuss, or Start Your Own

---

Subject: Re: Thought question: why is encrypted output routinely the same size as the input?
Date: Mon, 12 Oct 1998 13:45:54 +0100
From: Mok-Kong Shen <mok-kong.shen@stud.uni-muenchen.de>
Message-ID: <3621FA02.9C7D2394@stud.uni-muenchen.de>
References: <6vj3hfjii1@nnrp1.dejanews.com>
Newsgroups: sci.crypt
Lines: 63

dianelos@tecapro.com wrote:
>
> Mok-Kong Shen <mok-kong.shen@stud.uni-muenchen.de> wrote:
> > dianelos@tecapro.com wrote:
> > > produce and transmit noise R
> > > compute and transmit C = Encrypt(T xor R)
> >
> > You use noise often times as seed to a PRNG to obtain its output R
> > and XOR the plaintext T with R. (The phrase 'noise R' above seems to
> > be 'output R of PRNG using some noise as seeds'.) You encrypt (T XOR R)
> > to C and send both R and C. So you double the volume of transmission.
> > My point is that R is available to the analyst and this is
> > undesirable.
>
> Yes and no. Suppose a new random R is produced with each encryption.
> Then this simple method at least stops all *chosen* plaintext
> attacks. R is known after the encryption but an attacker can
> only choose T before the encryption. This method does not stop
> chosen ciphertext attacks, where the attacker can specify both R
> and C.
>
> > So I think one of your sentences above probably should read:
> >
> > use some noise N via PRNG to generate R and transmit N
> > compute and transmit C = Encrypt(T XOR R)
> >
> > But then I think one should transmit not N but U = Encrypt(N).
> >
>
> Here is what I want: A method that uses ciphers as a primitive
> and stops all known plaintext or ciphertext only attacks. I don't
> see why such a method cannot exist; if it turns out that such a
> method does exist and is practical it would be extremely important
> for many security systems where speed is not critical.

I don't yet fully understand. Isn't that your

```
        produce and transmit noise R
```

and my

```
           use some noise N via PRNG to generate R and transmit N
```

are functionally (without considering security) equivalent? In your case R is simply known to the analyst. In my case if the PRNG is not known (say there is a bunch of these to be chosen) R is unknown to him, or else transmit U = Encrypt (N) as I suggest. In the second case the volume of transmission is reduced.

As to the topic of chosen plaintext attack I think it is the reference point that is essential. You said that the attack is impossible because R is known after encryption. Your attack refers to your 'Encrypt'. But if you consider 'XOR with R' to be an encryption function 'Encrypt1' (even though in your case R happens to be known to the analyst) then one sees that the attack is a chosen plaintext attack against the product cipher 'Encrypt1'+'Encrpyt'. Whether this is possible or not possible is, I think, dependent upon the strength of this product cipher. If you think that is impossible, then it should be all the more true in the alternative scheme I suggested.

Could we perhaps discuss a little bit more?

M. K. Shen

Subject: Re: Thought question: why is encrypted output routinely the same size as the input?
Date: Mon, 12 Oct 1998 21:11:55 GMT
From: dianelos@tecapro.com
Message-ID: <6vtrar$7lq$1@nnrp1.dejanews.com>
References: <3621FA02.9C7D2394@stud.uni-muenchen.de>
Newsgroups: sci.crypt
Lines: 98

In article <3621FA02.9C7D2394@stud.uni-muenchen.de>,
  Mok-Kong Shen <mok-kong.shen@stud.uni-muenchen.de> wrote:
> dianelos@tecapro.com wrote:
> >   Here is what I want: A method that uses ciphers as a primitive
> >   and stops all known plaintext or ciphertext only attacks. I don't
> >   see why such a method cannot exist; if it turns out that such a
> >   method does exist and is practical it would be extremely important
> >   for many security systems where speed is not critical.
>
> I don't yet fully understand. Isn't that your
>
>   produce and transmit noise R
> and my
>
>   use some noise N via PRNG to generate R and transmit N
>
> are functionally (without considering security) equivalent? In your
> case R is simply known to the analyst. In my case if the PRNG is
> not known (say there is a bunch of these to be chosen) R is unknown
> to him, or else transmit U = Encrypt (N) as I suggest. In the second
> case the volume of transmission is reduced.

O.K. Let us suppose we have two encryption boxes, each of which includes a true random generator. For each plaintext T_i, the first box produces random R_i and outputs R_i as well as E(T_i xor R_i). The second box produces random N0 and outputs E(N0) - then for each plaintext it transmits E(PRNG(N_i) xor T_i).

Now, the first box cannot be attacked by a chosen plaintext attack. Even if the attacker can discover the secret key K used by E by choosing only *one* plaintext processed by E, it is impossible to make certain what data will be processed by E because each time it gets mixed with a new random R_i. The same goes for an attack that works if the attacker can choose any condition for a sufficient large number of blocks. By having the box mix one random bit with each plaintext bit the attacker has zero control over the data block that will be processed by E.

I think that, in principle at least, the second box can be attacked with chosen plaintext. First let us suppose without loss of generality that the same PRNG is always used. (If there is a choice of several, repeat the attack several times.) Every PRNG displays some kind of regularity that can be used in an attack. An extreme example, but sufficient in our discussion about principle, is that every PRNG loops back, say after M iterations. Then PRNG(N_i) = PRNG(N_(i+M)). Now an attacker can choose T_i and T_(i+M) with a particular difference which will be preserved after XORing it with the identical PRNG value. Thus a differential attack is in principle possible against the second box.

Now I agree that this discussion about principle may not be too important in practice. Suppose somebody would prove that for a particular E the following method cannot be attacked by *any* imaginable chosen plaintext attack:

```
        C1_i = E( R_i )
        C2_i = E( T_i or R_i )
```

Then we would have a quite practical, provably secure method, albeit one that doubles the size of the ciphertext. After all, chosen plaintext is the most intrusive method of attack - if that attack will not work in principle, neither will known plaintext or ciphertext only attacks.

Now if space or transmission speed is a limiting factor, then the true random variable R could be replaced by short PRNG sequences based on true random seeds N. For example a method, that produced a new random seed every 8 blocks and used it to produce a good pseudo-random R_i for the next 8 encryptions, would appear very secure too and would increase ciphertext size only in 12.5%. Any weakness in the PRNG would have to be exploited in only 8 blocks. The same method injecting noise every 100 blocks would increase ciphertext size only 1% and still look very good. This last method is what you propose.

Other possibilities exist. If E can encrypt large blocks another possibilitiy would be to use Terry's random fields. One would get an advantage if that random field were used as the seed to a PRNG and the resulting sequence were XOR-ed to the rest of the data block. For example, suppose E can encrypt 128 byte blocks:

For each 112 byte plaintext T_i, generate true random 16 byte N_i, use it as a seed to a good PRNG and produce 112 bytes of pseudo random R_i, and finally encrypt the 128 byte concatenation of N_i with (T_i xor R_i). Space inefficiency is now about 12.5% and security is probably much greater. Only, this method as well as all its variants can at least in principle be attacked by a chosen plaintext attack. On the other hand it is an open question how this method of "internal randomization" compares to the method of "external randomization" described above, which has the same space inefficiency of 12.5%.

--
http://www.tecapro.com
email: dianelos@tecapro.com

----------== Posted via Deja News, The Discussion Network ==----------
http://www.dejanews.com/ Search, Read, Discuss, or Start Your Own

---

Subject: Re: Thought question: why is encrypted output routinely the same size as the input?
Date: Tue, 13 Oct 1998 16:06:39 +0100
From: Mok-Kong Shen <mok-kong.shen@stud.uni-muenchen.de>
Message-ID: <36236C7F.3D87CBAF@stud.uni-muenchen.de>
References: <6vtrar$7lq$1@nnrp1.dejanews.com>
Newsgroups: sci.crypt
Lines: 35

dianelos@tecapro.com wrote:

> O.K. Let us suppose we have two encryption boxes, each of which
> includes a true random generator. For each plaintext T_i, the
> first box produces random R_i and outputs R_i as well as E(T_i xor
> R_i). The second box produces random N0 and outputs E(N0) - then
> for each plaintext it transmits E(PRNG(N_i) xor T_i).

You are right. A PRNG can hardly compete with a real TRNG. (There are some difficulties of obtaining and employing really good TRNGs though, in my humble opinion.)

> Now I agree that this discussion about principle may not be too
> important in practice. Suppose somebody would prove that for a
> particular E the following method cannot be attacked by *any*
> imaginable chosen plaintext attack:
>
> C1_i = E(R_i)
> C2_i = E(T_i or R_i)
>
> Then we would have a quite practical, provably secure method,
> albeit one that doubles the size of the ciphertext. After all,
> chosen plaintext is the most intrusive method of attack - if that
> attack will not work in principle, neither will known plaintext or
> ciphertext only attacks.

With the assumption previously of the availability of a TRNG I think one could do something practically useful by avoiding (or at least effectively reduce the probability of) the analyst being able to know the correspondence between C1_i and C2_i, like sending them through different channels or intermixing the bytes of both streams in some way. (BTW, one could certainly choose different algorithms for the two encryptions.)

M. K. Shen

Terry Ritter, his current address, and his top page.

Last updated: 1999-01-19

Ritter's Comments on The One Time Pad

```
From: ritter@io.com (Terry Ritter)
Newsgroups: sci.crypt,alt.security,alt.security.pgp,comp.security.pgp.discuss
Subject: Re: Bob spills the beans about real cryptography
Date: Tue, 25 Nov 1997 20:38:37 GMT
Lines: 47
Message-ID: <347b3391.971888@news.io.com>
References: <347aec1a.9997385@nntp.ix.netcom.com> <65etku$2ab$1@news.ox.ac.uk>
<347b0e35.18728269@nntp.ix.netcom.com> <65f6ab$e44$1@news.ox.ac.uk>


On 25 Nov 1997 18:44:27 GMT, in <65f6ab$e44$1@news.ox.ac.uk> in
sci.crypt patrick@gryphon.psych.ox.ac.uk (Patrick Juola) wrote:

>[...]
>*IF* the OTP is used properly (with all the caveats that we know so
>well by now), it's unconditionally secure.  However, the OTP is a
>very difficult system to use properly, and the chance of a catastrophic
>key-management failure is too high to neglect in most situations.

This is the conventional view, which I now believe is false.

The OTP which is "unconditionally secure" is *not* the realized OTP
which is used in practice, but instead the *theoretical* OTP which has
an ideal random keying source.  Since that source does not exist in
reality, the "unconditionally secure" OTP also does not exist in
reality.

There is another level of difference here before we get to a
conventional stream cipher, and that is the ever-increasing "entropy"
in the output of the physically-random generator used for OTP keying
material.  But, again, we cannot quantify this, and especially we
cannot guarantee it:  Should there come a time when the generator
produces *less* "entropy" than we expect, and the plaintext has *more*
"entropy" than we expect, the plaintext will not be protected: it will
"leak" information (even if just a fractional bit).  Clearly, any such
system could not be *unconditionally* secure, even if *generally*
secure in practice.

All of this comes before we face the difficulty of generating,
transporting, and keeping the keying material in absolute security.

Part of the problem here is that the words "one-time" serve to confuse
the issue.  We *can* attain one-time-ness; we *cannot* attain the
theoretical-class randomness which is required by the cipher described
by the name "one-time pad."

I claim that the "unconditionally secure" OTP is a *theoretical*
concept, and its proof is confined to theory.  A *realized* "OTP" is
*not* really an OTP at all, in that same sense.  The seeming advantage
of the OTP over all other ciphers is illusion rather than reality.

---
Terry Ritter   ritter@io.com   http://www.io.com/~ritter/
The Crypto Glossary:  http://www.io.com/~ritter/GLOSSARY.HTM
```

```
From: ritter@io.com (Terry Ritter)
Newsgroups: sci.crypt
Subject: Re: ONE TIME PAD?
Date: Tue, 25 Nov 1997 22:40:25 GMT
Lines: 53
Message-ID: <347b53a7.2307534@news.io.com>
References: <3479ae8d.1381598@news.io.com> <64ugau$8a4$1@news.ox.ac.uk>
<64uqcq$rs7@newsops.execpc.com> <64v67g$r7i$1@news.ox.ac.u
<65egu7$kv1$1@news.ysu.edu> <wtshaw-2511971427260001@207.101.116.48>


On Tue, 25 Nov 1997 14:27:26 -0600, in
<wtshaw-2511971427260001@207.101.116.48> in sci.crypt
wtshaw@itexas.net (W T Shaw) wrote:

>In article <65egu7$kv1$1@news.ysu.edu>, an096@yfn.ysu.edu (David A. Scott)
>wrote:
>
>> In a previous article, ritter@io.com (Terry Ritter) says:
>> >
>> >"It is possible to design *unbreakable ciphers*.  To do so, the key
>> >must be randomly selected (i.e., each key must have the same chance of
>> >being chosen) and used only once.  Furthermore, the length of the key
>> >must be equal to or greater than the length of the plaintext to be
>> >enciphered."

While Scott's article has not arrived here yet, I think it important
to note that this was not *me* talking; this was me selecting a
*quote* from Meyer and Matyas.  This quote is their particular view of
the one-time pad, and emphasizes the need for randomness.  Later
quotes showed how central the concept of randomness was to the OTP
security proof.  I didn't go back to Shannon, because I wanted views
of that work which had presumably matured over time.


>There is a big difference between absolutely unbreakable and *functionally
>unbreakable*.  The latter would not necessarily require either of the
>above.  Meanwhile, other desired characteristics of ciphers could be
>included.  Now, this is no excuse for accepting bad design, as the primary
>goal is to twart the cracker in any system to the extent that he gives
>up.  If you do that, the system is equivalent to the absolutely
>unbreakable one in effect.

My point in entering this thread is to take on what I perceive as the
fundamental basis for contention: the delusion that a "one-time pad"
is mathematically-proven to be absolutely secure, and therefore is
better than any other cipher.  Once one buys into this delusion, the
issues become things like: "Why would anyone use anything *other* than
an OTP," "If other ciphers are good, why don't they have proofs like
an OTP," "If your favorite cipher is so good, prove it," and "So what
if I have to transport an OTP key; once I do, the cipher is
unbreakable."  Then we go round and round, as we have been.

The problem is in the original assumption that a realized OTP *is* a
mathematically-proven unbreakable cipher; this is false.  Only the
*theoretical* OTP is proven unbreakable.  And the theoretical OTP is
just an *idea*, an unachievable goal.

---
Terry Ritter   ritter@io.com   http://www.io.com/~ritter/
The Crypto Glossary:  http://www.io.com/~ritter/GLOSSARY.HTM
```

```
From: ritter@io.com (Terry Ritter)
Newsgroups: sci.crypt,alt.security,alt.security.pgp,comp.security.pgp.discuss
Subject: Re: Bob spills the beans about real cryptography
Date: Wed, 26 Nov 1997 19:56:31 GMT
Lines: 91
Message-ID: <347c7e4f.10318738@news.io.com>
References: <347aec1a.9997385@nntp.ix.netcom.com> <65etku$2ab$1@news.ox.ac.uk>
<347b0e35.18728269@nntp.ix.netcom.com> <65f6ab$e44$1@news.ox.ac.uk>
<347b3391.971888@news.io.com> <david-2511971427460001@lax-ca69-58.ix.netcom.com>
<wtshaw-2511972153060001@207.101.116.63>


On Tue, 25 Nov 1997 21:53:06 -0600, in
<wtshaw-2511972153060001@207.101.116.63> in sci.crypt
wtshaw@itexas.net (W T Shaw) wrote:

>In article <david-2511971427460001@lax-ca69-58.ix.netcom.com>,
>david@sternlight.com (David Sternlight) wrote:
>
>> In article <347b3391.971888@news.io.com>, ritter@io.com (Terry Ritter) wrote:
>> >There is another level of difference here before we get to a
>> >conventional stream cipher, and that is the ever-increasing "entropy"
>> >in the output of the physically-random generator used for OTP keying
>> >material
>>
>> Is there any evidence that a random generator based on cosmic ray impacts
>> has any serial weakness?

I've never seen any serious proposal for this, and it takes some sort
of serious proposal to start to identify problems.  Personally, I
would expect that any such machine on the surface of the Earth would
be somewhat directional, and thus show some sort of daily and perhaps
yearly pattern.  Further, if we use some sort of Geiger-Muller tube,
it necessarily has response and quench time which will hide any other
arrival in this same time period.  And the data rate is probably low
at best.  Certainly we expect a Poisson distribution of periods
between pulses, which then must be processed into a flat distribution.



>The cost David, the cost, remember the cost.  Better to measure raindrops
>in Florida or weigh sand grains.  A detector that might pretend to measure
>cosmic ray impacts is measuring byproducts from collisions of cosmic rays
>with other particles, and is therefore subject to noise from other
>sources.--International Classroom, circa 1959.
>
>Mere radiation detecting, since particles are emitted from far from
>critical masses in relatively random fashion, would make a better choice.

David's article not having shown up yet locally, let me just make a
few comments that probably will not be very satisfying.

In really-random generators, there is a history of people having their
favorite noise sources.  Some people like thermal noise, some like the
emission of sub-atomic particles, others have other ideas.  Once
having decided on a noise source, there is a history of people doing
increasingly deep analyses of the result, often requiring continuing
modification or even re-engineering of the noise-sensing machinery.
Normally, the author gives us a usable system, but few if any put
their system forth as being "provably random."

The rain idea (which should be ideal shot noise!) seems attractive,
but then we have to wait for rain.  If we sprinkle our own, then we
probably can't trust the results.

I have considered using a sheet of partially-conductive rubber in a
rainstorm.  The rubber should compress under a raindrop impact, which
would deliver a noticeable voltage pulse (when driven by a current).
But when we get down to it, aren't there larger and smaller raindrops?
And when a large raindrop impacts, does this not splatter new false
"raindrops," thus hiding real (small) raindrops?  And if we *do* get
some large ones, doesn't the mere presence of the added water mass (as
it rolls away) now tend to hide small real raindrops?  And what about
wind, and rain at an angle?  And on and on.

This same process of analysis proceeds through many different sources
of noise and detection processes.  Thermal noise has its own problems,
and it takes extensive amplification at wide bandwidth and good
sampling to attain what seem like good random results.  But thermal
noise is tiny indeed, and we also get random-like results from
structured signals (e.g., music) and digital noise (device switching)
and power noise (switching power supply pulses).  Yet none of these
things actually *is* random, so how can we *prove* that they (or
something else) are not present?

There are a couple of vignettes about all this on my pages at:

   http://www.io.com/~ritter/RES/RNGMACH.HTM#McKean87

Also see:

   http://www.io.com/~ritter/RES/RNGMACH.HTM#Rand55

If anyone knows of any other published accounts of randomness-sensing
machinery, please let me know and I will try to include them in my
next page update.

---
Terry Ritter   ritter@io.com   http://www.io.com/~ritter/
The Crypto Glossary:  http://www.io.com/~ritter/GLOSSARY.HTM
```

```
From: ritter@io.com (Terry Ritter)
Newsgroups: sci.crypt,alt.security,alt.security.pgp,comp.security.pgp.discuss
Subject: Re: Bob spills the beans about real cryptography
Date: Thu, 27 Nov 1997 16:39:36 GMT
Lines: 34
Message-ID: <347da215.3074506@news.io.com>
References: <347b0e35.18728269@nntp.ix.netcom.com> <65f6ab$e44$1@news.ox.ac.uk>
<347b3391.971888@news.io.com> <65gs7u$jjr$1@news.ox.ac.uk> <347C42B1.FF6@medit3d.com>
<MPG.ee6d272a579961b989683@news.ccinet.net>


On Thu, 27 Nov 1997 00:44:28 -0600, in
<MPG.ee6d272a579961b989683@news.ccinet.net> in sci.crypt
grizz@the.bears.den (Grizz) wrote:

>[...]
>I've been reading these arguments about OTP's and randomness for some
>time now and you people seem to be missing two important points. First
>there is no such thing as true random numbers and second OTP's do not
>need true random number sequences to be unconditionally secure.

At least for me the issue has not been conditional versus
unconditional security, but instead, PROOF of security.

The basis for the PROOF of OTP security is the theoretical-class
randomness used in a theoretical OTP.  So if we want that same proof
to apply to a realized OTP, we must PROVE that our real RNG has
theoretical-class randomness in practice.  But such proof seems
impossible.

Instead of viewing the crypto landscape as being partitioned between
the proven OTP (and friends) versus unproven everything else, we now
see that the actual partition is between proven theoretical systems
and unproven realized systems of all kinds, including any realized
OTP.

I find this a more reasonable and more satisfactory view of
cryptography.

---
Terry Ritter   ritter@io.com   http://www.io.com/~ritter/
The Crypto Glossary:  http://www.io.com/~ritter/GLOSSARY.HTM
```

```
From: ritter@io.com (Terry Ritter)
Newsgroups: sci.crypt
Subject: Re: ONE TIME PAD?
Date: Mon, 01 Dec 1997 19:17:03 GMT
Lines: 56
Message-ID: <34830d22.5777581@news.io.com>
References: <3473aaa9.54902385@nntp.ix.netcom.com>
<wtshaw-2011970015400001@207.101.116.44> <34747e42.16121902@nntp.ix.netcom.com>
<6520ko$4b8$1@news.ox.ac.uk> <3474a93f.27126095@nntp.ix.netcom.com>
<652dbi$76d@news.microsoft.com> <EK0sxF.5yK.0.sheppard@torfree.net>
<3476d8b2.875829@nntp.ix.netcom.com> <wtshaw-2311970126420001@207.101.116.46>
<34785b2d.10001110@nntp.ix.netcom.com> <347943D6.167E@medit3d.com>
<34798f8b.5552654@nntp.ix.netcom.com>


On Mon, 24 Nov 1997 14:38:51 GMT, in
<34798f8b.5552654@nntp.ix.netcom.com> in sci.crypt
rcktexas@ix.netcom.com (R. Knauer-AIMNET) wrote:

>On Mon, 24 Nov 1997 10:07:34 +0100, Colin Dooley <colin@medit3d.com>
>wrote:
>
>>[...]
>>The theory (and practice) shows that these "breakable" methods
>>are secure enough even for military purposes. Why complicate
>>your life with all the problems caused by one time pads?
>
>Simple - until I see how these systems are provably secure, I will be
>forced to fall back on the only 100.000% unbreakable system, the OTP.

This seems a simple statement of the fundamental issue in this thread,
which of course Bryan Olson has confused in his past few postings.

The issue is "PROVABLE security."  Clearly, Bob assumes that a
realized one time pad *is* provably secure, and that anything else is
less.


>Can you tell me of a system that in provably secure to 99.999%, and if
>so would you trust extremely valuable information with it - like
>something that involves your own life? IOW, if the adversary cracks
>your code, you are instantly a dead person.

The classic information-theoretic proof of security for a one time pad
requires theoretical-class perfect randomness.  This is probably not
achievable, and certainly not provable.  We cannot hope to test every
possible relationship which might show non-randomness.  Any particular
relationship can only be measured statistically, and even a 99.999%
probability that a relation does not exist is not *proof*.

So if we go by the classic proof, the only provably secure one time
pad is a *theoretical* one time pad, which is only good for sending
theoretical data (or perhaps theoretically sending real data).  This
proof does not apply to a real system.

As soon as the one time pad is used in practice, it becomes a stream
cipher with a huge key.  Now, perhaps one could take on the particular
characteristics of a particular random number machine and show these
sufficient for security in practice, which might be some sort of
"proof."  That would still be better than most systems, but only
relevant here if such were actually achieved.

The classic proof simply does not apply.  The blanket statement that a
realized one time pad is "100.000% unbreakable" is just false.

---
Terry Ritter   ritter@io.com   http://www.io.com/~ritter/
The Crypto Glossary:  http://www.io.com/~ritter/GLOSSARY.HTM
```

```
From: ritter@io.com (Terry Ritter)
Newsgroups: sci.crypt,alt.security,alt.security.pgp,comp.security.pgp.discuss
Subject: Re: Bob spills the beans about real cryptography
Date: Mon, 01 Dec 1997 18:21:54 GMT
Lines: 28
Message-ID: <3483003a.2473022@news.io.com>
References: <347b0e35.18728269@nntp.ix.netcom.com> <65f6ab$e44$1@news.ox.ac.uk>
<347b3391.971888@news.io.com> <65gs7u$jjr$1@news.ox.ac.uk> <347C42B1.FF6@medit3d.com>
<347F1FF0.805F734@munich.netsurf.de>


On Fri, 28 Nov 1997 20:48:00 +0100, in
<347F1FF0.805F734@munich.netsurf.de> in sci.crypt Joachim Durchholz
<joachim.durchholz@munich.netsurf.de> wrote:

>[...]
>If you want a source of *real* random numbers, take some
>quantum-mechanical device. Like some radioactive substance and a Geiger
>counter, calibrated so that the probability of an event per (say)
>millisecond is 50% (you'd have to take the decrease in radioactivity
>into account though).

This seems to be a popular suggestion lately.  The latest Cryptologia
[Oct 1997, 21(4): 351] provides a little more handwaving:  The source
would be a thorium-impregnated mantle for a Coleman lantern.  A
commercial PC Geiger Counter would produce a pulse and sample an 8-bit
value from a counter running at 1 MHz or better.

My guess is that the practical problems would involve the lack of very
short periods between pulses, and reducing correlations due to short
periods.  "Entropy" could be condensed by hashing, which should give
very satisfactory practical results, but with no guarantees.

---
Terry Ritter   ritter@io.com   http://www.io.com/~ritter/
The Crypto Glossary:  http://www.io.com/~ritter/GLOSSARY.HTM
```

```
From: ritter@io.com (Terry Ritter)
Newsgroups: sci.crypt,alt.security,alt.security.pgp,comp.security.pgp.discuss
Subject: Re: Bob spills the beans about real cryptography
Date: Mon, 01 Dec 1997 18:25:17 GMT
Lines: 38
Message-ID: <348300e5.2643978@news.io.com>
References: <3471bc2f.11666034@nntp.ix.netcom.com>
<3474a68f.26438907@nntp.ix.netcom.com> <654c4h$o77$1@gannett.math.niu.edu>
<3475f84b.36668836@nntp.ix.netcom.com> <65cdme$aql$1@gannett.math.niu.edu>
<347a396d.49042068@nntp.ix.netcom.com> <3n2it8qjx.fsf@kmac.terisa.com>


On 24 Nov 1997 21:56:34 -0800, in <3n2it8qjx.fsf@kmac.terisa.com> in
sci.crypt EKR <ekr@terisa.com> wrote:

>[...]
>Loosely speaking, the probability that a system as a whole will be
>compromised is
>1-(1-Pcryptosystem)(1-Pusers)
>
>where
>Pcryptosystem=the probability that the system will be compromised through
>some inherent weakness (e.g. cryptanalylis, brute force)
>Pusers=the probability that the system can be compromised through the
>users, e.g. by rubber hose cryptanlysis, key reuse, weak keys,
>bribery, etc.
>
>Now, we know that for the OTP, Pcryptosystem is zero. I.e.
>it's unconditionally secure.

This is the conventional view, which I now believe is false.

The "unconditionally secure" OTP is a *theoretical* entity, which uses
a handwaved random generator with assumed perfect theoretical
randomness properties.

If we realize an OTP for actual use and desire a similar proof of
"unconditional security," we must *also* realize a random-number
source with guaranteed theoretical-class randomness.  We do not have
such generators.

We do have good sources of really random numbers.  But we cannot prove
with absolute certainty that they are indeed *perfectly* random.

---
Terry Ritter   ritter@io.com   http://www.io.com/~ritter/
The Crypto Glossary:  http://www.io.com/~ritter/GLOSSARY.HTM
```

```
From: ritter@io.com (Terry Ritter)
Newsgroups: sci.crypt,alt.security,alt.security.pgp,comp.security.pgp.discuss
Subject: Re: Bob spills the beans about real cryptography
Date: Mon, 01 Dec 1997 20:13:47 GMT
Lines: 129
Message-ID: <34831a40.9136451@news.io.com>
References: <3471bc2f.11666034@nntp.ix.netcom.com>
<3474a68f.26438907@nntp.ix.netcom.com> <654c4h$o77$1@gannett.math.niu.edu>
<3475f84b.36668836@nntp.ix.netcom.com> <65cdme$aql$1@gannett.math.niu.edu>
<347a396d.49042068@nntp.ix.netcom.com> <3n2it8qjx.fsf@kmac.terisa.com>
<348300e5.2643978@news.io.com> <3lny4hp7c.fsf@kmac.terisa.com>


On 01 Dec 1997 10:51:19 -0800, in <3lny4hp7c.fsf@kmac.terisa.com> in
sci.crypt EKR <ekr@terisa.com> wrote:

>ritter@io.com (Terry Ritter) writes:
>
>> On 24 Nov 1997 21:56:34 -0800, in <3n2it8qjx.fsf@kmac.terisa.com> in
>> sci.crypt EKR <ekr@terisa.com> wrote:
>>
>> >[...]
>> >Loosely speaking, the probability that a system as a whole will be
>> >compromised is
>> >1-(1-Pcryptosystem)(1-Pusers)
>> >
>> >where
>> >Pcryptosystem=the probability that the system will be compromised through
>> >some inherent weakness (e.g. cryptanalylis, brute force)
>> >Pusers=the probability that the system can be compromised through the
>> >users, e.g. by rubber hose cryptanlysis, key reuse, weak keys,
>> >bribery, etc.
>> >
>> >Now, we know that for the OTP, Pcryptosystem is zero. I.e.
>> >it's unconditionally secure.
>>
>> This is the conventional view, which I now believe is false.

>Since my argument was that even in the face of an unconditionally
>secure cryptosystem, it was still possible for a conventional
>cryptosystem to be stronger as a total system, a fortiori, a
>this applies to a non-unconditionally secure OTP.

You said: "Pcryptosystem=the probability that the system will be
compromised through some inherent weakness," and "for the OTP,
Pcryptosystem is zero. I.e. it's unconditionally secure."

But no real one time pad has such a proof.  And if you are going to
argue about imaginary things, it scarcely seems right to claim that
*my* argument is "just semantics" (see below).


>> The "unconditionally secure" OTP is a *theoretical* entity, which uses
>> a handwaved random generator with assumed perfect theoretical
>> randomness properties.
>>
>> If we realize an OTP for actual use and desire a similar proof of
>> "unconditional security," we must *also* realize a random-number
>> source with guaranteed theoretical-class randomness.  We do not have
>> such generators.
>>
>> We do have good sources of really random numbers.  But we cannot prove
>> with absolute certainty that they are indeed *perfectly* random.

>You're just arguing semantics.

I disagree.


>There's always a theory part and
>an implementation part. Theoretically an OTP is secure. It
>can be implemented insecurely.

The reality is much worse than this.  In fact, an OTP *cannot* be
implemented with *proven* security.


>Due to the limitations of
>the physical world, it cannot be implemented completely securely.

Anything can be implemented poorly.  The problem is that, even when
implemented as well as possible, the realized one time pad *still*
does not have the theoretical property which is ascribed to it, or at
least we cannot prove it does.  Such proof is the whole basis for this
conversation.


>Conventional cryptosystems aren't even perfectly secure
>in theory, though.

This is not universal.  In fact, it has gotten disturbingly popular to
write articles which claim *proven* security, when in fact that
security is based on unattainable theoretic components.

However, even if a conventional cryptosystem *did* have a theoretical
analog which was proven perfectly secure, if that proof did not apply
to the real system, the proof would be totally irrelevant as support
for any idea that the real cryptosystem was better than any other.


>This isn't that important since the practical
>considerations of using an OTP are so different from those with
>a conventional cryptosystem.

The issue to the newbies we see here every month or so is PROVABLE
security in a REAL system.  But since the security proof for one time
pads does not survive the transition from theory to reality, their
whole argument has a false basis.

The cryptographic community bears some amount of blame for this by not
making the issue clear from the outset.  The *theoretical* quality of
the proven secure one time pad is something which should be in every
text, but, alas, is not.


>But, imagine two systems which were similar in usage, but whereas
>one was provably secure (to some level) and the other wasn't.
>For instance, imagine we knew factoring was hard, then a
>system like RSA which was provably as hard as factoring would
>be superior to RSA (which hasn't been proved to be as hard
>as factoring) provided that they could be used in similar ways.

First, Bob has not been arguing this.

Next, in this example we do *not* have a system which is proven secure
in theory, as would correspond to the theoretical one time pad.

Last, proof is proof.  A proven-in-theory system is fine as long as we
use it for sending theoretical data.  But if the proof does not
survive the transition to a real system, it is no proof at all for
that system.  A proof which does not apply is a just a delusion; it
has absolutely no relevance.  It does *not* give us any confidence in
the real system.  There is *no* advantage to using a real one time pad
based on the proof of absolute security in a theoretical one time pad.


---
Terry Ritter   ritter@io.com   http://www.io.com/~ritter/
The Crypto Glossary:  http://www.io.com/~ritter/GLOSSARY.HTM
```

```
From: ritter@io.com (Terry Ritter)
Newsgroups: sci.crypt,alt.security,alt.security.pgp,comp.security.pgp.discuss
Subject: Re: Bob spills the beans about real cryptography
Date: Tue, 02 Dec 1997 04:45:06 GMT
Lines: 110
Message-ID: <34839228.2906161@news.io.com>
References: <3471bc2f.11666034@nntp.ix.netcom.com>
<3474a68f.26438907@nntp.ix.netcom.com> <654c4h$o77$1@gannett.math.niu.edu>
<3475f84b.36668836@nntp.ix.netcom.com> <65cdme$aql$1@gannett.math.niu.edu>
<347a396d.49042068@nntp.ix.netcom.com> <3n2it8qjx.fsf@kmac.terisa.com>
<348300e5.2643978@news.io.com> <34830e17.20494489@nntp.ix.netcom.com>


On Mon, 01 Dec 1997 20:15:22 GMT, in
<34830e17.20494489@nntp.ix.netcom.com> in sci.crypt
rcktexas@ix.netcom.com (R. Knauer-AIMNET) wrote:

>On Mon, 01 Dec 1997 18:25:17 GMT, ritter@io.com (Terry Ritter) wrote:
>
>>This is the conventional view, which I now believe is false.
>
>Each time I begin to think we have these OTP/randomness issues under
>control, another problem pops up.
>
>>The "unconditionally secure" OTP is a *theoretical* entity, which uses
>>a handwaved random generator with assumed perfect theoretical
>>randomness properties.
>
>Please give us your definition of "perfect theoretical randomness".

Nope.  This is not my definition to give.  Perfect randomness is
constructed by a mathematical handwave in the proof that the
theoretical one time pad is absolutely secure.  I previously gave
several of those proof descriptions.  None quantified what randomness
was.  This is theoretical math: they don't have to say what randomness
is; they just imply what it is not.  If someone says "I can predict
thus and so" they just say, "Nope, the sequence is absolutely random
and not predictable," and that is that.


>I have been working on the basis that "perfect theoretical randomness
>for purposes of crypto" is that the next bit in the random stream is
>not predictable with a probability significantly greater than 1/2.
>Granted that is not as stringent as "perfect theoretical randomness"
>in general, but it is my understanding that it is both necessary and
>sufficient for the purposes of cryptography.

This *is* a stringent requirement.  It is my understanding that this
means that, given knowledge of *all* previous bits, using *every*
possible test, the next bit *still* cannot be predicted better than
chance.  This is another theoretical math handwave definition.  It
describes a result, rather than a way to get that result.  It cannot
be verified by testing, because it is impossible to enumerate every
possible test so each can be checked.

The usual mathematical approach (I believe) is to prove that if one
could make such a prediction in the context of the defined system,
then one could also solve some hard problem.  But I don't think this
is your approach.


>>If we realize an OTP for actual use and desire a similar proof of
>>"unconditional security," we must *also* realize a random-number
>>source with .  We do not have such generators.
>
>Not knowing your definition of "guaranteed theoretical-class
>randomness" I cannot comment whether such RNGs exist or not. I would
>argue that crypto-grade RNGs can be achieved in QM.

There is little argument that good randomness exists at the quantum
level, regardless of whether it is fundamental or the complex result
of hidden variables.  That randomness probably will not have the
desirable flat distribution we would like, however, and so must be
processed even in the best possible case.

But to use quantum randomness it must be detected, and this normally
requires some sensitive device which is an imperfect human construct.
For example, a Geiger-Muller tube takes some time to avalanche, and
much longer to clear out.  Events in that period are lost.  So not
only do we lose short-period events, but it also not possible to know
the precise time between the lost event and the next event, which is
the theoretical random value we want.  There are no doubt other
deviations to handle.  I know of no way to perfectly sense quantum
events.

No doubt we would want to hash what randomness we detect.  But doing
this is both beneficial and dangerous.  It is beneficial because it
tends to "concentrate" randomness within the hash value, which is what
we want.  It is dangerous because even non-random inputs look random
when hashed.  This means that faulty detection could result in
predictable values which still look random on the surface.


>>We do have good sources of really random numbers.
>
>The criterion above for crypto-grade randomness does not require
>perfection, just "significantly greater than 1/2".

But 0.5 *is* perfection.  To the extent that any delta above that can
be detected, only perfection counts.  It is hard to imagine that there
possibly can be a delta value below which we have (proven) security in
practice.  If you are pinning your hopes on "significantly," I doubt
it was meant to be taken that way.


>>But we cannot prove
>>with absolute certainty that they are indeed *perfectly* random.
>
>Can we prove with some measurable certainty that they are
>*cryptographically* random?

If you mean by "cryptographically random" the phrase "cannot predict
the next bit with probability greater than 0.5," my guess would be
that any such generator would be difficult to produce, and impossible
to prove.  It might work, and you might think it works, but you could
never know for sure.  Which is right back where we were, of course.

---
Terry Ritter   ritter@io.com   http://www.io.com/~ritter/
The Crypto Glossary:  http://www.io.com/~ritter/GLOSSARY.HTM
```

```
From: ritter@io.com (Terry Ritter)
Newsgroups: sci.crypt
Subject: Re: ONE TIME PAD?
Date: Tue, 02 Dec 1997 01:32:26 GMT
Lines: 102
Message-ID: <34835f43.4158034@news.io.com>
References: <3473aaa9.54902385@nntp.ix.netcom.com>
<wtshaw-2011970015400001@207.101.116.44> <34747e42.16121902@nntp.ix.netcom.com>
<6520ko$4b8$1@news.ox.ac.uk> <3474a93f.27126095@nntp.ix.netcom.com>
<652dbi$76d@news.microsoft.com> <EK0sxF.5yK.0.sheppard@torfree.net>
<3476d8b2.875829@nntp.ix.netcom.com> <wtshaw-2311970126420001@207.101.116.46>
<34785b2d.10001110@nntp.ix.netcom.com> <347943D6.167E@medit3d.com>
<34798f8b.5552654@nntp.ix.netcom.com> <34830d22.5777581@news.io.com>
<34832b72.28009795@nntp.ix.netcom.com>


On Mon, 01 Dec 1997 21:34:27 GMT, in
<34832b72.28009795@nntp.ix.netcom.com> in sci.crypt
rcktexas@ix.netcom.com (R. Knauer-AIMNET) wrote:

>On Mon, 01 Dec 1997 19:17:03 GMT, ritter@io.com (Terry Ritter) wrote:
>
>>The classic proof simply does not apply.  The blanket statement that a
>>realized one time pad is "100.000% unbreakable" is just false.
>
>To use your terminology, I meant  theoretically unbreakable.
>
>I, like you, have doubts about "practical randomness" in the classical
>realm, even with physical systems, because we cannot certify that the
>output stream is "practically random", even for crypto.

My position is that we cannot certify that any stream is *perfectly*
random, as required for the usual OTP proof.  This places the realized
OTP back in the body of all realized ciphers, instead of making it a
special case.

On the other hand, I would think that it *should* be possible to in
some sense "certify" that a random stream "seems like" it ought to be
random *enough* for practical use.  This is just no *proof*.


>I would like to believe, however, that there are certain kinds of
>randomness that will suffice for crypto, albeit not with 100.000%
>certainty.

Once we give up the holy grail of absolute certainty, we can then
begin to discuss practical cryptography in general, and stream ciphers
in particular.

But this idea of "percentage of certainty" is strange.  It would seem
to me that the only way to get such a value is to know more than we
can possibly know.  Nor would "percentage of randomness" be better,
for if there were some maximally-random stream, we could just send
that, or easily correct any stream we produce.  But this conflicts
with the idea of unpredictability.

The usual approach to describing cipher strength is to build a cipher
which contains within it multiple puzzles, each having independent
strength, but which cannot be solved independently.  We collect every
approach we can think of to start such solutions, and estimate the
effort required in each case.  We then report the minimum effort as
the cipher strength.  This of course depends upon seeing an
appropriate attack, and evaluating it properly, which are both major
problems in cryptography.  I have no doubt that this whole process may
seem an incredible way to support the main feature which cryptography
should provide -- strength -- but there it is.

Since the ultimate source of strength in a realized OTP is the really
random generator, any analysis of the system must be heavily involved
with the specific design and analysis of the actual generator itself.
We cannot handwave our way out of defining that before we analyze it.



>If I were to build a classical H/W RNG that put out what I
>believed to be a random stream - one that you could not predict the
>next bit with a probability significantly greater that 1/2 - would
>that not qualify as an "almost perfectly" random OTP?

I suppose that is a good definition, but we must understand that this
does not mean that only do analysis at bit level, or just think about
the next bit.  Quite often really random RNG's will have some amount
of correlation between local bits.  And there may well be
circumstances where correlation would be conditional on prior values
or even sequences.

It is also quite common in noise-based equipment that extraneous
electronic noise (e.g., repetitive pulses from a switching supply)
could be interpreted by the equipment as real noise and so produce
correlations with widely-spaced bits.  And if the fundamental basis
for collecting values is bytes (words, etc.) instead of bits, then a
bit-level analysis may not find correlations which naturally exist.


>If not, what attack would you implement to decrypt my ciphertext if I
>used it?

Just because we cannot use the information-theoretic proof of security
on a realized OTP does not mean that we now have a new attack:  A
realized one time pad may well be secure in practice.

But, in this still theoretical context, the cryptanalyst would
"simply" *find* some relationship in the sequence, and then use that
to acquire some amount of information from the ciphertext.

In practice, a properly used real one time pad with an extensively
analyzed (and post-processed!) really random generator is likely to be
very secure.  So the real issues in practice are those which have more
risk, the issues of assuring proper use, and guaranteeing the security
of the pad itself.  Which is what everybody was basically getting at
before I dropped in.

---
Terry Ritter   ritter@io.com   http://www.io.com/~ritter/
The Crypto Glossary:  http://www.io.com/~ritter/GLOSSARY.HTM
```

```
From: ritter@io.com (Terry Ritter)
Newsgroups: sci.crypt
Subject: Re: Real Random Number Generator Algorithm Publication
Date: Tue, 16 Dec 1997 21:34:09 GMT
Lines: 34
Message-ID: <3496f37b.1914332@news.io.com>
References: <01bd0023$5eb3d0e0$c901a8c0@comm> <3485E90C.124F@helsbreth.org>
<348603a1.43586684@nntp.ix.netcom.com> <EKqHAo.69G.0.sheppard@torfree.net>
<3488b2d3.18863924@nntp.ix.netcom.com> <348D9411.60842FCE@stud.uni-muenchen.de>
<348da790.28113815@nntp.ix.netcom.com> <348E0262.41C6@medit3d.com>
<348ea4c1.5211103@nntp.ix.netcom.com> <1997121203112074952@zetnet.co.uk>
<34913aa2.5163344@nntp.ix.netcom.com> <1997121400380274952@zetnet.co.uk>
<3493ed9b.5440753@nntp.ix.netcom.com> <wtshaw-1612971423480001@207.101.116.61>


On Tue, 16 Dec 1997 14:23:33 -0600, in
<wtshaw-1612971423480001@207.101.116.61> in sci.crypt
wtshaw@itexas.net (W T Shaw) wrote:

>In article <3493ed9b.5440753@nntp.ix.netcom.com>, rcktexas@ix.netcom.com wrote:
>>[...]
>> >I should also have mentioned that OTPs don't provide any protection against
>> >message modification attacks,
>>
>> I simply do not see that - please elaborate.
>
>Say you send the same message to three people, with three pads, differing
>only slightly.  Learn the contents of one and solve all the pads, and hope
>the sender is dumb enough to reuse a pad.

Perhaps a better example would be in sending the same message to two
people, using two pads.  The Opponents intercept one (as ciphertext)
and hold it, then penetrate physical security of the recipient of the
other message to get the plaintext.  Since they assume or know the
same message was sent in both instances, they now know the plaintext
of the first message, and also the content of the first pad, even
though that has neither been "attacked" nor physically exposed.  Then
The Opponents can *change* the content of the intercepted message to
read as they want, encipher it under the one-time pad they now know,
and send it along, to produce the results they want.  Nobody will
question that message, since everyone knows that a one-time pad system
is absolutely secure beyond any possible question.

---
Terry Ritter   ritter@io.com   http://www.io.com/~ritter/
The Crypto Glossary:  http://www.io.com/~ritter/GLOSSARY.HTM
```

Terry Ritter, his current address, and his top page.

Last updated: 1998-01-16

More sci.crypt Discussions

Is Triple-DES Stronger than DES?

Is this *really* proven?

- 1994-11-08 Robert Egendorf: What algorithms besides IDEA and 3xDES are extremely difficult to break?
- 1994-11-11 Terry Ritter: we do **not** *know* that either IDEA or Triple-DES *are* difficult to break
- 1994-11-12 Ken Pizzini: Monoalphabetic substition ciphers form a group. DES does not.
- 1994-11-13 Robert Egendorf: Has anyone else evaluated the Cloak2 cipher? What tests has it been subjected to? What is Mr. Ritter's background in cryptography?
- 1994-11-15 Terry Ritter: (replying to Ken) Since this is irrelevant in context, I am at a loss. (The issue is an attack on the overall permutation. Although this attack is impractical, it demonstrates that at least one attack does exist which is not complicated by additional block cipherings. Thus, any reasoning which implies that three cipherings *must* be stronger than one, is simply wrong.)
- 1994-11-15 Terry Ritter: (replying to Robert) Reputation is irrelevant to reality. The issue is the argument.
- 1994-11-15 David A. Wagner: It does seem reasonable to believe that triple DES is stronger than DES.
- 1994-11-16 Ken Pizzini: (replying to Terry) the proof that DES is not a group tells us that the keyspace of DES does get enlarged by composition.
- 1994-11-18 Terry Ritter: (replying to David) If I had a workable attack I could defeat your argument, but requiring me to have and disclose such an attack before you will move to a stronger cipher must defeat your own security. It is instead necessary to anticipate attacks, instead of simply responding to attacks as they become disclosed. Attacks may exist and we may not know them, and yet, to provide good crypto, we must defeat them anyway. Thus we must assume that such attacks exist.
- 1994-11-19 Bohdan Tashchuk: (replying to Terry) Spending 3x the compute cycles of single-DES to encrypt information today gives us an algorithm that most experts feel is much more than three times as secure. Spending 10x or even 100x the compute cycles isn't an unreasonable thing to ask.
- 1994-11-21 Greg Rose: (replying to Ken) I'm sorry, but the last statement (the proof that DES is not a group tells us that the keyspace of DES does get enlarged by composition) is not strictly true

It is not clear to me that Ken understood that I had proposed an attack on the overall permutation. Under any particular key, a block cipher is nothing more than Simple Substitution on a block. No matter how many levels there are, the overall transformation is still a block-wide Simple Substitution.

While a codebook attack is generally impractical, it puts lie to the claim that Triple anything is *necessarily* stronger than Single anything. Groupiness has nothing to do with it.

Although Bohdan may be willing to pay any cost for crypto he thinks secure, in my experience, this is an unusual position. On the contrary, network managers are under *extreme* pressure to keep up. Even though communication capabilities continue to rise, the demands for increased bandwidth rise much faster. Dreams and desires can always outstrip technical progress.

Network managers often see crypto as a necessary evil, an overhead to the expense of communication. While individuals may have plenty of compute power, network managers currently cannot keep up as it is, and so are strongly motivated to have fast crypto, or none at all.

Modified RC4 Becomes a Dynamic Substitution

Putative RC4 improved.

- 1994-11-11 Farid F. El-Wailly: I'd like to suggest a modification of RC4-like algorithms that would make them a little more resistant to the key re-use problem.
- 1994-11-14 Terry Ritter: Mr. El-Wailly appears to have re-invented the concept of Dynamic Substitution, which is protected by U.S. Patent 4,979,832. That said, I don't see Dynamic Substitution as a solution to the problem of key re-use. A better way is to have a random message key in every message
- 1994-11-15 Peter K. Boucher: I coded this up based on a description under a thread about improving RC4. It runs a little faster than optimized DES.
- 1994-11-15 Stefan Lucks: (responding to Farid) The cryptanalysis of two xored plaintexts is not trivial.
- 1994-11-15 Stewart Strait: (responding to Stefan) I believe you're mistaken.
- 1994-11-16 Padgett 0sirius: (responding to Terry) Near as I can tell that covers any forward substitution scheme in which the final transformation is a function of a cyclical algoritm which include the previous block as a component.
- 1994-11-16 Steve O'Neill: (responding to Stewart) from an operational point of view, changing keys for every transmission is an absolute requirement
- 1994-11-18 Terry Ritter: (responding to Padgett) it is a non-trivial exercise to try and define technical mechanisms precisely. Patentese may fail to do so, but compare it to ordinary writing and one can see certain advantages
- 1994-11-19 J.M. Kelsey: (responding to Farid) I don't think the modification you suggest would make it safe to re-use the key
- 1994-11-19 Stewart Strait: (responding to Steve) XORing one unknown message with another is _not_ equivalent to a one-time pad unless 'unknown' means 'so unknown that all possible messages are roughly equally likely'.

SAFER K-64

What is it?

- 1994-11-01 John Kelsey: The SAFER K-64 algorithm was designed by James Massey for Cylink, and was presented at the Cambridge Security Workshop in December 1993
- 1994-11-01 Serge Vaudenay a kown plaintext attack will be presented in next december against SAFER with N=6 in which the log_45 is replaced by a random permutation. This attack does not work with the log_45, but it shows both the weakness of the general shape of SAFER and the strength of the particular design chosen by James Massey.
- 1994-11-30 Andrew Haley: The idea of using the FFT-like permutations for rapid diffusion is rather nice, but the choice of the S-box is a bit of an enigma
- 1994-12-01 Michael Roe: I have a cut-down version of SAFER that works on 4 bit nibbles rather than 8-bit bytes, and I can prove that its round functions generate the full symmetric group
- 1994-12-01 Serge Vaudenay: In this paper, it is shown that a necessary condition for the strength of the substitution S is that the least significant bit is unbiased
- 1995-03-23 Richard DeMoliner:As I did for IDEA I developed a software package for the encryption algorithm SAFER. This package is now publicly available and the source code belongs to the public domain.

Generalized Feistel Networks

A new idea?

- 1995-04-02 Ralph Brown: Feistel ciphers are based on repeated rounds . . . for the two halves A and B of a block. This idea can be generalized to the N parts of a block. For N subblocks in a block, a minimum of N rounds are required to process each subblock uniformly, at which point every subblock of the output depends on every subblock of the input.
- 1995-04-03 Stewart Strait: If the mixing functions are linear, we get a simple form of the Hill System
- 1994-04-03 Bruce Schneier: The function f does not have to be invertable at all; the Feistel structure takes care of the invertability. Matt Blaze and I also tried to generalize the Feistel construction, but in such a way as to preserve the use of a noninvertable function f.We presented our strawman construction, MacGuffin, at the Leuven Algorithms Workshop last December, and it was immediately broken.
- 1995-04-03 Ralf Brown: Fair enough.
- 1995-04-04 Bruce Schneier: you can look at SHA as a block function turned into a hash function with a Davies-Meyers-like feedforward function.) Haval has a similar construction, as do (more or less) MD4 and MD5. The attack was based on our choice of f, which was ripped out of DES with little thought about how the changes might affect it; the attack didn't hve anything to do with the structure.

Terry Ritter, his current address, and his top page.

Last updated: 1995-12-27

Simon's Braided Stream Cipher

William "Alain" Simons proposal for using a really-random stream to select between multiple data channels on a bit-by-bit basis. One or more of the channels might also be really-random, in which case new key material can be transported to the far end as a side-effect.

Of course, if we are allowed to expand the ciphertext by 2x, virtually any cipher can transport key material in a separate message.

- 1991-06-13 Relayed from William "Alain" Simon: The Braided Stream cipher
- 1991-06-15 der Mouse: der responds negatively
- 1991-06-17 der Mouse: der responds positively
- 1991-06-18 Alain Simon: Alain says the Braid is not "new"
- 1991-06-23 Alain Simon: Alain responds to Jerry Leichter. (Apparently a message was lost in which Jerry claims that a known-plaintext attack will converge into the correct key. But if the key is "really random" that would seem to be little help. Part of the problem here is an inability to pin down what the design really is.)
- 1991-06-23 Dan Boyd: Dan supports Jerry.
- 1991-06-23 Doug Gwin: Doug says that the scheme would be stronger if multiple keys could produce the same ciphertext.
- 1991-06-24 John Nagle: John enters the fray
- 1991-06-24 Alain Simon: Alain responds to Dan, basically saying that the key is "really random" so that recovering it with a known-plaintext attack is not much of an advantage.
- 1991-06-24 Dan Bernstein: Bernstein contributes his wisdom
- 1991-06-24 Peter Wayner: Peter jumps in
- 1991-06-24 Doug Gwin: Doug jumps on Bernstein
- 1991-06-24 Dan Bernstein: Bernstein points out that a key-bit is used up for every ciphertext bit. This means that the Braid apparently cannot add to the amount of key by transporting keying material in the second channel.
- 1991-06-24 Alain Simon: Alain responds to Doug, saying that different keys *could* produce the same ciphertext
- 1991-06-24 Alain Simon: Simon says: "take a break"
- 1991-06-26 Alain Simon: Alain proposes a scheme to increase the amount of transported key material
- 1991-06-26 Ken Shirriff: Ken thinks braiding is weaker than XOR
- 1991-06-27 Arthur Rubin: Arthur feels that braiding is a little stronger than XOR
- 1991-06-27 Alain Simon: Alain responds to Ken I
- 1991-06-28 Alain Simon: Alain responds to Ken II
- 1991-06-29 der Mouse: der responds to Dan
- 1991-07-02 Alain Simon: Alain responds to Arthur I
- 1991-07-02 Alain Simon: Alain responds to Arthur II: "Eating pretzels." Braiding *is* weaker than XOR.
- 1991-07-12 David Seal: David jumps in
- 1991-07-15 Alain Simon: Alain responds to David
- 1991-07-17 David Seal: David responds to Alain
- 1991-07-18 Alain Simon: Alain recapitulates the wandering scheme: "Braid Crumbs"
- 1991-07-21 Alain Simon: Alain responds to David
- 1991-07-21 Alain Simon: Alain responds to himself
- 1991-07-23 David Seal: David responds to Alain
- 1991-07-24 Terry Ritter: After showing unusual forbearance, Terry can resist no longer and finally responds with some results from balanced combiner theory
- 1991-07-24 Alain Simon: Alain responds to Terry

[Here at least one Simon and Ritter exchange was lost.]

- 1992-08-18 Ross Anderson: Ross feels that the Braid was "demolished" in: Anderson, R. 1990. Solving a Class of Stream Ciphers. *Cryptologia* 14(3): 235-238. But the referenced article attacks multiplexed RNG's, whereas the Braid multiplexes data.
- 1992-11-05 Alain Simon: Alain points out that the Braid can contain multiple different ciphertexts, a characteristic which could be useful in forced contact with law-enforcement.

Terry Ritter, his current address, and his top page.

Last updated: 1995-10-31

Differential Cryptanalysis

A *Ciphers By Ritter* Page

What the heck is Differential Cryptanalysis anyway?

Contents

- 1999-01-11 Frank Gifford: "You can create a list of all possible [4 bits] input differences and see what the output differences are allowed after going through a round."
- 1999-01-13 Scott Fluhrer: "Lets see if I can go through of an example of how DC would be used against this cipher some detail...."

```
Subject: Re: Differential Cryptanalysis???
Date: 11 Jan 1999 15:50:21 -0500
From: giff@eng.us.uu.net (Frank Gifford)
Message-ID: &LT;77do6d$q5t@trebuchet.eng.us.uu.net>
References: &LT;nHqm2.4035$TO5.107922@ptah.visi.com>
Newsgroups: sci.crypt
Lines: 29

In article &LT;nHqm2.4035$TO5.107922@ptah.visi.com>,
Michael A. Greenly &LT;newsgreenly@caidesign.com> wrote:
>
>    I've been trying to get a handle on differential crytptanalysis for
>the last week or two but seem to have run into a road block of sorts.  I
>think I understand most of it but there's one part the seems to elude
>me.  I don't understand how a right pair suggests a key?

Here's what I can suggest based on the educational cipher you describe on
your web page.

Notice that the input and outputs to the Left halves are affected only
by the output of the second round function.  So create some pairs of
plaintexts which differ by a known amount in their input, encrypt, and
see how the output changes.

By seeing the difference between the two left halves, you know how the
output from the second round has changed.

You can create a list of all possible [4 bits] input differences and see
what the output differences are allowed after going through a round.

Try a couple pairs by hand and see what you get.

-Giff


--
giff@uu.net       Too busy for a .sig
```

```
Subject: Re: Differential Cryptanalysis???
Date: Wed, 13 Jan 1999 03:13:21 GMT
From: Scott Fluhrer &LT;sfluhrer@ix.netcom.com>
Message-ID: &LT;77h2kv$lps@sjx-ixn9.ix.netcom.com>
References: &LT;nHqm2.4035$TO5.107922@ptah.visi.com>
Newsgroups: sci.crypt
Lines: 97

In article &LT;nHqm2.4035$TO5.107922@ptah.visi.com>,
        "Michael A. Greenly" &LT;newsgreenly@caidesign.com> wrote:

>
>    I've been trying to get a handle on differential crytptanalysis for
>the last week or two but seem to have run into a road block of sorts.  I
>think I understand most of it but there's one part the seems to elude
>me.  I don't understand how a right pair suggests a key?

Lets see if I can go through of an example of how DC would be used
against this cipher some detail (and, forgive me if I go through detail
you already know -- this tiny explination might help someone else who
doesn't know as much as you).

First of all, to apply DC, you need to find differentials that hold with
(relatively) high probability.  In case you didn't know, a differential
is a pair of plaintexts whose bit differences flow through the cipher
in a predictable way.

To find such high probability differentials, you go through the sbox,
and look for differences in inputs that produce consist differences in
the output.  Going through your sbox, I (actually, a computer program:
I'm lazy) find these high probability differences:

If the input of the sbox is xored by A (1010), the output bits change
(is xored by) 4 (0100) with probability 50%

If the input of the sbox changes by 5, the output changes by 9 with
probability 37.5%

If the input of the sbox changes by 8, the output changes by 3 with
probability 37.5%

Now, using these regularities in the sbox, we can assemble high-probability
differentials:

  If the input to the cipher changes by A4 (10100100), then the internal
data after the first half-round changes by 0A with probability 50% (because
the output of the sbox will change by 4 with probability 50%, and everything
else is linear).  Given that, the second half-round changes by A0 with
probability 100% (because the input to the sbox is that same in both cases).
Given that, the third half-round changes by 4A with probability 50%, giving
an input-to-output differential with probability 0.5*1*0.5 = 0.25.

Or, pictorially, the differential looks like (and, refer back to
http://www.pinenet.com/~mgreenly/twisted.gif for what goes through
the sboxes):

  A   4
   \ /
    X      Probability 0.5
   / \
  0   A
   \ /
    X      Probability 1
   / \
  A   0
   \ /
    X      Probability 0.5
   / \
  4   A

Now, the attacker encrypts pairs of plaintexts which differ by A4, and examine
the corresponding ciphertext.  If the ciphertexts do not differ by 4A, he
discards them (actually, he could in this case fruitfully analyze them, but
not in a real cipher, which may have differentials with probability 0.00001)
If he does find such a pair, then he knows that either:

  - It was a coincidence (unlikely in this case, but a real possibility with a
    real cipher).  You make sure by finding several probable diffentials; or

  - This is a manifestation of the differential.  In which case, he knows that:

      - The input to the first sbox was either 1, 3, 4, 6, 9, B, C or E (the 8
        inputs that will cause this differential to occur).  And, since he knows
        the input bits (which are the plaintext bits), he knows the first
        half-round subkey bits are in one of 8 settings.  In effect, he learns
        one bit from the subkey

      - The input to the third sbox was either 1, 3, 4, 6, 9, B, C or E (the 8
        inputs that will cause this differential to occur).  And, since he knows
        the input bits (which are the ciphertext bits), he knows the third
        half-round subkey bits are in one of 8 settings.  In effect, he learns
        another bit from the subkey

So, by trying an average of 4 plaintext pairs, he gets 2 key bits.  Using
different differentials (eg. the 59->95 differential, which occurs at
probability 0.14), he whittles down the key to a point where he can
exhaustively search the rest of the key bits.


--
poncho
```

Terry Ritter, his *current address*, and his *top page*.

Last updated: 1999-02-20

What is a "Group" in Block Cipher Analysis?

A *Ciphers By Ritter* Page

We assume that Triple-DES is stronger than plain DES because DES is not a group. So what does that mean?

Contents

- 1998-12-15 Patrick Juola: "...the relevant property here is that a group is 'closed' under a particular operation."
- 1998-12-15 Andrew Haley: "A group (G, *) is a set G with a binary operator * with the following properties...."
- 1998-12-15 John Savard: "If DES encryption _were_ a group... then double-DES, or triple-DES, or quintuple-DES, would just be regular DES in disguise."
- 1998-12-15 Bryan Olson: "The pure Feistel structure of DES generates only even permutations, so the group generated by DES is a subgroup of the alternating group (which has half the elements of the symmetric group)."
- 1998-12-15 keaak@tgr.org: "A semigroup is a set S together with an associative binary operation, t : SxS----> S. Nothing is said of the existence of inverses, nor of an identity element under t...."
- 1998-12-16 John Pliam: "There are finite semigroups which are not groups." "However, there are no sub-semigroups of a finite group."
- 1998-12-26 fungus: "Imagine you encrypt some data twice with seperate keys K1 and K2 If there exists a single key K3 which can decrypt the data then the encipherment is a group."

```
Subject: Re: What is "group" means in block cipher ?
Date: 15 Dec 1998 09:26:46 -0500
From: juola@mathcs.duq.edu (Patrick Juola)
Message-ID: &LT;755rj6&49?$1@quine.mathcs.duq.edu>
References: &LT;36766E0C.F0BF70B2@sec-cse.sch.ac.kr>
Newsgroups: sci.crypt
Lines: 22

In article &LT;36766E0C.F0BF70B2@sec-cse.sch.ac.kr>,
Seung-Chul, Chae &LT;chae@sec-cse.sch.ac.kr> wrote:
>Someone said, DES encryption generates a semigroup and other encryption
>generates a group.
>What is "group" means ? it is a set of block ?


A group is a mathematical abstraction; the relevant property here is
that a group is "closed" under a particular operation.

Which is to say, if you do the operation once (with one parameter)
and then do the operation again (with another parameter), there's some
third parameter that could do it in one step.

Just as an example, the set of whole numbers is closed under addition.

If I add three to a number X (getting X + 3), and then add five to
the result, that's exactly the same as if I had added eight directly
to the original number.  So the operation pair (adding five, adding three)
is exactly the same as the operation (adding eight).

        -kitten
```

```
Subject: Re: What is "group" means in block cipher ?
Date: 15 Dec 1998 16:51:13 GMT
From: aph@cygnus.remove.co.uk (Andrew Haley)
Message-ID: &LT;756421$aet$1@korai.cygnus.co.uk>
References: &LT;36766E0C.F0BF70B2@sec-cse.sch.ac.kr>
Newsgroups: sci.crypt
Lines: 24

Seung-Chul, Chae (chae@sec-cse.sch.ac.kr) wrote:
: Someone said, DES encryption generates a semigroup and other encryption
: generates a group.
: What is "group" means ? it is a set of block ?

A group (G, *) is a set G with a binary operator * with the following
properties:


Associativity: a * (b * c) = (a * b) * c

There is an identity element I: a * I = I * a, for all a elem G.

For each a elem G there exists an inverse a ^ (-1) elem G,
a * (a^(-1)) = (a^(-1)) * a = 1


DES is not a group; that is the operation DES on a block does not have
these properties.

You ought to get a book on the mathematics behind cryptography before
proceeding any further.

Andrew.
```

```
Subject: Re: What is "group" means in block cipher ?
Date: Tue, 15 Dec 1998 17:16:54 GMT
From: jsavard@tenMAPSONeerf.edmonton.ab.ca (John Savard)
Message-ID: &LT;36768Dc4.875397@news.prosurff.com>
References: &LT;36766E0C.F0BF70B2@sec-cse.sch.ac.kr>
Newsgroups: sci.crypt
Lines: 21

"Seung-Chul, Chae" &LT;chae@sec-cse.sch.ac.kr> wrote, in part:

>Someone said, DES encryption generates a semigroup and other encryption
>generates a group.
>What is "group" means ? it is a set of block ?

If DES encryption _were_ a group - the term is taken from algebra -
which it is not,

then for any two keys x, y, there would exist a key z such that
encrypting any and all plaintext blocks, first with key x, then with
key y, exactly the same result could be obtained by just encrypting
once with key z.

If that were true, then double-DES, or triple-DES, or quintuple-DES,
would just be regular DES in disguise. Even though one would not know
what the key is that is equivalent, for someone not knowing the key,
it would be no harder to crack.

John Savard
http://www.freenet.edmonton.ab.ca/~jsavard/index.html
```

```
Subject: Re: What is "group" means in block cipher ?
Date: Tue, 15 Dec 1998 11:45:42 -0800
From: Bryan Olson &LT;first.last@xuptronicsx.com>
Message-ID: &LT;36768C66.C851EE5C@xuptronicsx.com>
References: &LT;36766E0C.F0BF70B2@sec-cse.sch.ac.kr>
Newsgroups: sci.crypt
Lines: 28


"Seung-Chul, Chae" wrote:
> Someone said, DES encryption generates a semigroup and other encryption
> generates a group.
> What is "group" means ? it is a set of block ?

I'm not really up on semigroups, but I understand that given a
finite set of elements, they're the same thing as groups.

DES, like all block ciphers, generates a group.  The elements of
the group are one-to-one functions from 64 bit blocks to 64 bit
blocks (equivalently, permutations of the set of 64 bit blocks).
Specifically, any permutation that can be generated as a composition
of DES encryptions (possibly with different keys) is in the set.
The group operation is function composition.

The pure Feistel structure of DES generates only even permutations,
so the group generated by DES is a subgroup of the alternating
group (which has half the elements of the symmetric group).  If DES
is any good at simulating random even permutations, it should
generate the alternating group.

A couple other posts answered the question of whether DES _is_ a
group, which is not the same as the one you asked about generating
a group.  A block cipher is a group iff each of the permutations
in the group it generates is equivalent to some single encryption.

--Bryan
```

```
Subject: Re: What is "group" means in block cipher ?
Date: Tue, 15 Dec 1998 23:06:04 GMT
From: keaak@tgr.org
Message-ID: &LT;756q2r$sht$1@news-2.news.gte.net>
References: &LT;36768C66.C851EE5C@xuptronicsx.com>
Newsgroups: sci.crypt
Lines: 37

On Tue, 15 Dec 1998 11:45:42 -0800, Bryan Olson
&LT;first.last@xuptronicsx.com> wrote:

>
>"Seung-Chul, Chae" wrote:
>> Someone said, DES encryption generates a semigroup and other encryption
>> generates a group.
>> What is "group" means ? it is a set of block ?

>I'm not really up on semigroups, but I understand that given a
>finite set of elements, they're the same thing as groups.

No, they're not the same thing.  A semigroup is a set S together with
an associative binary operation, t : SxS----> S.  Nothing is said of
the existence of inverses, nor of an identity element under t
(Although the latter is included in the definition in some texts).

A trivial example:  Let S={1,s} and define a 'multiplication' in the
obvious way except define s^2=s.  This defines a semigroup, and it's
finite.  To see it's not a group, try to find the inverse of s under
this multiplication.

Someone already defined a 'group' in this thread.

Mike


Decrypt keaak@tgr.org with ROT13 for email address.

NOTICE TO BULK EMAILERS:  Pursuant to US Code,
Title 47, Chapter 5,  Subchapter II,  227, any
and all nonsolicited commercial E-mail sent to
this address is subject to a download and archival
fee in the amount of $500 US. E-mailing denotes
acceptance of these terms.
```

```
Subject: Re: What is "group" means in block cipher ?
Date: 16 Dec 1998 02:02:48 GMT
From: John Pliam &LT;jpliam@ima.umn.edu>
Message-ID: &LT;36771FC8.524C073F@ima.umn.edu>
References: &LT;757ikk$1un$1@nntp1.ounet.ca>
   &LT;756q2r$sht$1@news-2.news.gte.net>
Newsgroups: sci.crypt
Lines: 61

jmc@mvcpu.org wrote:
> any pointer on a paper about that ? i know there is one paper about
> the proof that des is not a group, but i dont remember the title.

See refs below.


Bryan Olson &LT;first.last@xuptronicsx.com> wrote:
> "Seung-Chul, Chae" wrote:
> > Someone said, DES encryption generates a semigroup and
> > other encryption generates a group. What is "group" means ?
> > it is a set of block ?
>
> I'm not really up on semigroups, but I understand that given a
> finite set of elements, they're the same thing as groups.


There are finite semigroups which are not groups.  (An important
family of examples comes from finite automata whose state transition
diagrams are not a Cayley graphs.  The "syntactic monoid" of such an
automaton is the semigroup of partial functions induced on the state
space by words in the regular expression.  This semigroup is not
a group.)

However, there are no sub-semigroups of a finite group.  Thus the
requirement of decryption means that no block cipher can generate a
semigroup (which is not itself a group).

> If DES is any good at simulating random even permutations, it
> should generate the alternating group.


Wernsdorf [1] showed that, in fact, the one-round functions
of DES generates the full alternating group.  It can be inferred
that the 16-round functions do as well.

> A couple other posts answered the question of whether DES _is_ a
> group, which is not the same as the one you asked about generating
> a group.  A block cipher is a group iff each of the permutations
> in the group it generates is equivalent to some single encryption.


Yeah, I think things are sufficiently confusing and the results
are sufficiently spread out that a summary could be useful to the
beginner:

 o No block cipher can generate true semigroup.
 o No block cipher can  `be'' a true semigroup.
 o Any block cipher generates a subgroup of the symmetric group S_(2^n).
 o Any Feistel cipher generates a subgroup of A_(2^n). [2]
 o DES ``is'' not a group. [3]
 o DES generates the full alternating group, A_(2^n). [1]

John Pliam
pliam@ima.umn.edu
http://www.ima.umn.edu/~pliam

refs:


[1] R. Wernsdorf, "The One-Round Functions of DES Generate the
    Alternating Group", EUROCRYPT '92, pp. 99-112, 1993.
[2] S. Even & O. Goldreich, "DES-Like Functions can Generate the
    Alternating Group, IEEE inf. theo., IT-29:6, pp. 863-865, 1983.
[3] K.W. Campbell and M.J. Wiener, "DES is not a Group",
    CRYPTO "92", pp. 512-517, 1992.
```

```
Subject: Re: What is "group" means in block cipher ?
Date: Sat, 26 Dec 1998 09:04:19 -0100
From: fungus &LT;spam@egg.chips.and.spam.com>
Message-ID: &LT;3684B4A3.1A92ABE1@egg.chips.and.spam.com>
References: &LT;36766E0C.F0BF70B2@sec-cse.sch.ac.kr>
Newsgroups: sci.crypt
Lines: 21

Seung-Chul, Chae wrote:
>
> Someone said, DES encryption generates a semigroup and other
> encryption generates a group.
> What is "group" means ? it is a set of block ?

Imagine you encrypt some data twice with seperate keys K1 and K2

If there exists a single key K3 which can decrypt the data then
the encipherment is a group.

In DES this is not possible. You may find one key which will decrypt
one block of data, but it would be luck. You probably won't even be
able to decrypt other blocks of the same message.

--
&LT;\___/>
/ O O \
\_____/  FTB.
```

Terry Ritter, his *current address*, and his *top page*.

Last updated: 1999-02-20

The Value of Cryptanalysis

A *Ciphers By Ritter* Page

This huge conversation starts out with the article by Schneier. That article is controversial in various ways:

- the path to success in cryptography is cryptanalysis, and the path to that is academic publication.
- cryptanalysis is how we know the strength of ciphers,
- a cipher with an "impractical break" should be seen as weaker than ciphers without such a break.

These arguments bring out fundamental issues in cryptography which are generally assumed to have been resolved long ago, with the answers now obvious. See my response, my later response and someone else's response and math descriptions.

Contents

- 1998-10-17 Bruce Schneier: "Congratulations. You've just invented this great new cipher, and you want to do something with it. You're new in the field; no one's heard of you, and you don't have any credentials as a cryptanalyst. You want to get well-known cryptographers to look at your work. What can you do? "Unfortunately, you have a tough road ahead of you."
- 1998-10-18 George Barwood: "I disagree - some time ago I posted an algorithm to sci.crypt, and recieved a quick (and useful) analysis from David Wagner."
- 1998-10-18 Karl-Friedrich Lenz: "Probably Mr. Schneier intended to say 'not a second glance by professionals in scientific papers', which might be true. But the level of sci.crypt is not that low, and there seem to be quite a lot of people ready to have a swing at new ideas."
- 1998-10-18 Bruce Schneier: "You're right. There are exceptions to this. Agreed."
- 1998-10-18 Jon Haugsand: "Actually, wouldn't this be a good way to train oneself with cryptoanalyzing? Breaking amateur ciphers posted to the usenet?"
- 1998-10-19 Bruce Schneier: "Definitely. I think it's the best way. Not only do you get experience breaking ciphers, but you get some very easy ones to start on."
- 1998-10-17 W T Shaw: "A contrived obstacle course means being sure that few can finish, and more are discouraged from even trying."
- 1998-10-18 Lloyd Miller: "Bruce's religion makes a lot more sense to me than your's."
- 1998-10-18 Jay Holovacs: "If you can't break codes that are out there, why should anyone believe that you have an answer."
- 1998-10-18 W T Shaw: "In Bruce's work, there are sinful omissions and comissions, but the subject is so large that this would always be a surity in some form. To judge his character, we will see if he mentions in the future any things he has previously ignored and have been pointed out directly to him."
- 1998-10-18 dscott@networkusa.net: "I like your chemsitry example it fits well witht the load of stuff Bruce is trying to pass off."
- 1998-10-19 : "...cryptanalysis is a discipline of its own, and requires either considerable stamina or advanced mathematical skills. One does not quite need these qualifications to design a secure cipher, particularly if one is following your earlier advice and not ignoring the lessons of previous designs."
- 1998-10-19 Mark Tillotson: "Nonsense! How on earth can you claim to design a secure cipher if you are _incapable_ of distinquishing a weak cipher from a strong cipher???"
- 1998-10-22 John Savard: "...while a _knowledge_ of cryptanalysis is needed, actually being a cryptanalyst - actually being able to carry out, in full, the cryptanalysis of a difficult cipher, or being able to make theoretical contributions to the field - is not, strictly speaking, necessary...."
- 1998-10-22 W T Shaw: "Many imply that if you simply follow their rules for cipher construction, you need not do much of the analysis yourself."
- 1998-10-26 Bruce Schneier: "Many are wrong."
- 1998-10-25 W T Shaw: "...the AES process is *designed* as a big feedback mechanism, the quicker acting the better."
- 1998-10-26 Bruce Schneier: "Rah rah."
- 1998-10-26 cryptonews@my-dejanews.com: "This is not about crypto and security, it is rather becoming about Bruce Schneir BIG EGO and what he thinks the world should be."
- 1998-10-26 dscott@networkusa.net: "For a while I thought I was the only one intelligent enough to notice Mr B.S. is nothing but a big BLOWHART it seems that every one else was following him like a god."
- 1998-10-26 John Savard: "No, that is not at all true or fair." "Actually, if there were 10,000 amateur cipher designs published, the harm would be mainly to amateur cipher designers...."
- 1998-10-28 Gurripato (x=nospam): "If those 10.000 amateur cipher existed and were published, crypto vendors would start incorporating them into their products. How would the customers react when 9.990 of those ciphers are proved to be weak?"
- 1998-10-28 Terry Ritter: "This is a legitimate concern, but it applies to everything we have." "The problem is that we cannot measure the strength of a cipher. But that means *any* cipher, even the well-regarded ones."
- 1998-10-28 Patrick Juola: "This is untrue. It's fairly easy to come up with a measurement of the strength of a cypher...."
- 1998-11-02 Terry Ritter: "From the user's standpoint, an upper bound is *not* the strength, and is not even a useful estimate." "To the user, since we have *neither* the real strength, *nor* the lower bound, we have no useful measure of strength at all."
- 1998-11-02 Patrick Juola: "I have an upper bound, I insure against the lower bound being smaller than I envision, and the risk becomes Lloyd's."
- 1998-11-10 Terry Ritter: "When cryptanalysis identifies a practical break, it provides very useful information." "But most cryptanalysis does not do this, but instead produces yet another impractical break."
- 1998-11-10 John Savard: "...since differential, meet-in-the-middle attacks, etc., require enormous quantities of known plaintext, either it is not clear they invalidate a system for practical use...."
- 1998-11-10 Bruce Schneier: "To many of us, impractical breaks provide very useful information to judge between ciphers."
- 1998-11-11 Douglas A. Gwyn: "They provide information, which you may *choose* to use in judging, but that is not necessarily a rational choice. To be rational, its *relevance* to the functional criteria needs to be established."
- 1998-11-12 Terry Ritter: "Finding a successful attack certainly tells us that "strength" can be no higher than that attack. But it does not tell us what the strength really is. So the attack tells us *nothing* about the real strength of the cipher."
- 1998-11-03 Sandy Harris: "I think that for good ciphers, lower bounds on the resources required for most or all of those can be proved."
- 1998-11-03 dscott@networkusa.net: "...how much information is needed by the guy breaking to know if he his decoded the file."
- 1998-11-03 Mike McCarty: "This principle seems good to me."
- 1998-11-03 Terry Ritter: "...if it were practical to know lower bounds for these attacks, why would we ever see improved versions in the literature?"
- 1998-11-06 Bryan G. Olson; CMSC (G): "You've talked yourself into a bunch of nonsense."
- 1998-11-02 John Savard: "I feel, on the other hand, that this isn't a problem one *can* work on specifically."
- 1998-11-03 Terry Ritter: "The whole point of the actual use of cryptography is to *enforce* security. Without at least a minimum value for strength, the user has no guarantee -- or even a useful probability -- of that."
- 1998-11-03 John Savard: "I don't contradict your statement that this is a serious problem for cryptography... but if there is no realistic prospect of obtaining it... however badly we need it, is _still_ a waste of time." "...on a very high level, cryptanalysis can be divided into three types of operation:...."
- 1998-11-04 Douglas A. Gwyn: "It's nice to try to bring order to the subject, but the above is not complete."
- 1998-11-04 dscott@networkusa.net: "...sometimes the encryption program itself does not use or solve for the key that the method is based on."
- 1998-11-10 Terry Ritter: "It was suggested that cryptanalysis is the way users know the strength of their ciphers. That suggestion is false." "In reality, cryptanalysis only benefits *users* when their particular cipher is actually shown to be weak in practice *and* the user can switch to something else."

- 1998-11-10 John Savard: "...between two closely similar ciphers, the one protected against the impractical break but not otherwise different is likely to be stronger."
- 1998-11-04 Jerry Leichter: "There is no proof of security for key locks, combination locks, or any other means of providing physical security." "With any of the well-studied cipher systems out there today, it's unlikely that the mathematical structure of the cipher will be the weakest component of a real-world system in which it is embedded."
- 1998-11-06 Bruce Schneier: "Profoundly true."
- 1998-11-10 Terry Ritter: "But we can be more sure about these simple devices than the vast majority of far-more-complex ciphers." "The problem is these 'special understandings.' As long as we produce ciphers that admit new cryptanalysis, we cannot be sure of their true strength. If we cannot somehow confine or bound the unknown 'special understandings,' we will never have factual grounds to state that cipher strength is 'unlikely' to be the weakest part of the system."
- 1998-11-10 Jerry Leichter: "...there are *no* published attacks with any real-world significance - except for brute force relying on limited key spaces - against any of, say, DES, RC4, IDEA, or RSA." "You can explain this difference in three ways:"
- 1998-11-11 Douglas A. Gwyn: "More than three."
- 1998-11-11 Jerry Leichter: "Well, OK."
- 1998-10-18 dscott@networkusa.net: "Part of the NSA job is to keep the world in the dark about real ctypto. Think about it. What better way to do it than by creating crypto preists for people to whorship."
- 1998-10-19 Tim Bass: "Most of those whom have written strong ciphers did not write them without very significant research into the field."
- 1998-10-19 dscott@networkusa.net: "John you shouldn't try to confuse a Bruce Worshiper with facts."
- 1998-10-19 David Hamilton: "Has the USA NSA succeeded in keeping you in the dark about 'real crypto'?"
- 1998-10-19 dscott@networkusa.net: "Obviously you don't read all of crapola...."
- 1998-10-26 dscott@networkusa.net: "What I don't like is the spying on Americans for political reasons that will someday make what the Soviet Union had look like a dream of a long lost freedom."
- 1998-10-20 dscott@networkusa.net: "...if he had a contest it would be embarassing to have a rank ametur break it."
- 1998-10-20 Terry Ritter: "It is not the responsibility of the developers to go around and inform all the 'experts' through their chosen media outlet. Either they keep up, or they are not experts on what they have missed, and it's just that simple.
- 1998-10-22 Bryan G. Olson; CMSC (G): "I have to agree with Mr. Ritter on this one."
- 1998-10-22 dscott@networkusa.net: "...even if the name is removed I bet any one with have a brain could tell mine from Bruces and from Mr Ritter since we all 3 have different writting styles even if we all 3 write about the exact same subject."
- 1998-10-22 Mark Carroll: "What interests would the review panel have in choosing Bruce's paper over yours if yours is so much better?"
- 1998-10-23 dscott@networkusa.net: "...they may be aready attuned to his narrow closed style of thinking since the reveiwers most like got to there positions in the same way he did and they may not be any more cabable of objective thought than he is."
- 1998-10-22 Andrew Haley: "Why should anyone be bothered to read what you write if you can't be bothered to correct any of your mistakes?"
- 1998-10-22 Patrick Juola: "...for most major conferences, it's expected."
- 1998-10-22 Terry Ritter: "...this entire thread is a response to the original article by Schneier...." "...he clearly *did* imply that *something* prevents "unknowns" from publishing in conferences and workshops."
- 1998-10-23 Patrick Juola: "...if you're a total unknown, you probably won't get workshop invitations. You can, however, easily get into conferences *if* you can write a good enough paper...."
- 1998-10-26 Bruce Schneier: "...'hard' is not impossible."
- 1998-10-26 Terry Ritter: "...it is largely the lack of a broad and robust literature on breaks of all types which makes 'the newbie problem' as bad as it is. The process of selecting only good designs for the archival literature leaves us with little description of the bad ones, and less archived reasoning about their weaknesses. I claim we would be better off if every newbie cipher was presented and broken in the literature."
- 1998-10-27 dscott@networkusa.net: "Mr RItter I feel that Bruce is one of those self inflated people incapable of understanding your writting. He is afraid of real competition so will attempt to put it done with jokes and such...."
- 1998-10-26 John Savard: "But we can be very thankful he published his design."
- 1998-10-27 "Keith Lockstone": "This idea has been published before on sci.crypt...."
- 1998-10-23 dscott@networkusa.net: "REAL CRYPTO conferences should have executable program or functions where the input and output can be analysed and various real testing done on computers."
- 1998-10-22 W T Shaw: "You are confusing narrow mindedness with focus."
- 1998-10-23 Patrick Juola: "I suspect that his ability to master English will be a more vital asset for his eventual programming abilities."
- 1998-10-23 Andrew Haley: "...you can gain some idea of the level of a programmer's skill just by listening to them."
- 1998-10-26 Mok-Kong Shen: "I am however anyway convinced that if one has acquired sufficient proficiency in a foreign language, the difference between a foreign language and one's mother tongue disappears."
- 1998-10-27 fungus: "...there are some concepts which have a word in one language but not in another. Sometimes you find yourself arrive at the middle of a sentence wanting to use a word from the other language because no equivalent exists in the language you're speaking."
- 1998-10-27 Mok-Kong Shen: "That's why good translations of master pieces are rare."
- 1998-10-27 Patrick Juola: "French, for example, has no single word meaning 'shallow.'" "This does NOT, however, mean that the French don't understand the distinction between deep and shallow water, or even that they can't talk about it."
- 1998-10-25 W T Shaw: "Prejudice by language, life style, heritage, anything you want to throw in. Concentrating on style rather that substance is easy, and wrong."
- 1998-10-26 Bruce Schneier: "You can figure out the authors of some papers without the authors' names, but not all of them. You can easily figure out who is schooled in the mathematics of cryptography and who isn't."
- 1998-10-26 Bruce Schneier: "...even the conferences that referee papers anonymously don't publish design papers unless they are REALLY impressive."
- 1998-10-26 Bryan G. Olson; CMSC (G): "...here's how to really get a design published in the crypto lit: Find some new and interesting fact, develop a design that incorporates the result, then write a paper that presents both the theorem and the system."
- 1998-10-26 Bruce Schneier: I invite you to submit a paper, based on your patent #5,727,062 ('Variable Size Block Ciphers') to the 1999 Fast Software Encryption workshop. I believe it will be published." (Ed. Note: This public invitation was later retracted in private email./TFR)
- 1998-10-26 Bruce Schneier: "Please submit your good ideas to cryptography workshops. FSE and SAC are good places to start." "Statistical tests are not very meaningful. If you saw a cipher design that was accompanied by nothing other than statistical tests of randomness, wouldn't your snake-oil detector go off?"
- 1998-10-26 W T Shaw: "Statistics can measure more things than randomness."
- 1998-10-26 John Savard: "That's all Bruce was saying; statistics aren't enough - although specialized statistical tests, directly related to the possible forms of cryptanalysis that a cipher may face, can, of course, be very applicable."
- 1998-10-26 Terry Ritter: "We *never* know that a cipher is strong. Ever." "Now, we might 'consider' a cipher strong when all *our* guys have looked at it and found no break. But, quite frankly, the *other* guys have more training, more experience, more resources, more time, and they may even be smarter than our guys." "I claim it is more important to have many different ciphers than to have a few which are 'considered strong.' Why? Because we *can't* know how strong our ciphers *really* are to the other guy. But we *can* -- guaranteed -- make The Opponent pay dearly to keep up."
- 1998-10-27 John Savard: "This is something I basically agree with."
- 1998-10-28 Christopher Browne: "As far as I can tell, the only reasonably universal such language is that of mathematical notation."
- 1998-10-28 Terry Ritter: "I see nothing wrong with ordinary people making their own decisions on cryptography -- or anything else -- based on whatever information they wish to use."
- 1998-10-28 W T Shaw: "You could run the risk of producing some interference pattern in the combination of algorithms that could produce a poor result...."
- 1998-10-28 Terry Ritter: "While *possible*, in the context of structurally-different ciphers it is *extremely* unlikely."
- 1998-10-29 dscott@networkusa.net: "It is obvious that mixinf three different types of ciphers would be better than Triple DES...."

- 1998-10-29 Terry Ritter: "...in practice, most of the time, ciphers only need oppose direct technical attacks which are cheaper than bribery, and that will be a pretty weak attack. In that sense, weak ciphers may be less of a problem than having a single fixed cipher that might be cryptanalyzed once and used to expose everybody." "Since we can't know what NSA can do, I think it can be a waste of time to worry about it."
- 1998-10-29 Tim Bass: "From an intellectual perspective, I've read nothing which is remotely enlightening from this entire thread."
- 1998-10-29 Andrew Haley: "How do you expect a female cryptographer feels when told to conduct herself like a gentleman?"
- 1998-10-30 Douglas A. Gwyn: "...some of my best friends are women -- but I wouldn't want my sister to marry one!"
- 1998-10-30 dscott@networkusa.net: "That is a sexist statement if I ever saw one."
- 1998-10-30 Tim Bass: "'Please Little Boys, Be Nice, Stop Fighting and Play Together!'"
- 1998-10-30 Tim Bass: "Restraint from harsh and offensive speech would make sci.crypt a much more positive experience for everyone, IMHO."
- 1998-10-31 dscott@networkusa.net: "Are you for real."
- 1998-10-31 Douglas A. Gwyn: "...trying to change the language to force one's political views on the world is sickening."
- 1998-10-30 dscott@networkusa.net: "I liked your answer...."
- 1998-10-29 Jerry Leichter: "Not only is it extremely unlikely - it would be a direct indication that *both* of the ciphers involved were weaker than expected."
- 1998-10-29 Bruce Schneier: "Indeed. You cannot prove that a cascade of several ciphers is stronger than any individual cipher, but is seems reasonable that it is the case."
- 1998-10-29 W T Shaw: "Reason requires consideration of details."
- 1998-10-29 Bruce Schneier: "I know of various efforts to look at the AES submmissions with respect to different attacks, but I have never heard of anyone looking at the possibilty of short cycles or group structure."
- 1998-10-29 W T Shaw: "You can only mix a few things in so many ways in a fixed length block until your ciphertext is identical with one of your previous plaintexts."
- 1998-10-29 Terry Ritter: "If these 'short cycles' are just those which naturally appear in random permutations, surely a large block is a prescription to make it unlikely that we could ever find one, or encounter one by chance."
- 1998-10-30 dscott@networkusa.net: "...the Paul Onion attack for a choosen plain test file if allowed shows that if cycle length known you can taylor an attack against a pure iterating cipher."
- 1998-10-30 Jerry Leichter: "What we need to know is that the short cycles - all of whose members correspond to "weak keys" of a sort - amount to only an insignificant fraction of the group."
- 1998-10-30 Terry Ritter: "...a random permutation of reasonable size should not have this difficulty."
- 1998-11-02 Jerry Leichter: "That's easy to prove...."
- 1998-10-29 W T Shaw: "And, we find that the effective keylength is somewhat less than 3 times DES."
- 1998-10-29 ssimpson@hertreg.ac.uk: "The best we can hope to do is use our complete arsenal of analysis tools to prove that a cipher is insecure. If it fails to succumb to these tools then it is not _proven_ to be secure, but it indicates that a degree of faith can be placed in the cipher."
- 1998-10-30 Terry Ritter: "Concluding that a cipher which has not been shown weak is therefore strong is surely incorrect reasoning. So the cipher may be weak. And if the cipher *is* weak, we surely would be fools to have faith in it, no matter how much analysis was done previously."
- 1998-10-30 ssimpson@hertreg.ac.uk: "But we have to have faith in one (or possibly more) block ciphers. Rather than pick this cipher at 'random' it is surely better to pick the a block cipher that has been subjected to and resisted all known attacks."
- 1998-10-30 Terry Ritter: "I guess faith is about the only thing we *can* have. But that's religion, not science. We may use a cipher, but we *cannot* trust it." "...I have come to believe that it may be more important to use a multiplicity of ciphers -- accepting their possible weaknesses -- than to use a single cipher -- and accepting its possible weakness."
- 1998-10-30 Douglas A. Gwyn: "...I can exhibit the design for a block cipher that is demonstrably secure according to the rules of the game, although it wouldn't be *practical*."
- 1998-10-30 Bruce Schneier: "While it is certainly possible to, in theory, give a proof of security that does not also prove that P != NP, most formulations of such a proof--which, of course, does not exist--hinge on proving P != NP."
- 1998-10-31 Douglas A. Gwyn: "I don't think *any* block ciphers have anything to do with P?=NP."
- 1998-11-02 John Savard: "...if a proof that P=NP is interpreted as indicating there are no mathematical problems that get really intractable to solve, compared to the effort required to verify the solution, then that would seem to affect everything - even if the application to secret-key ciphers would still be awkwards."
- 1998-11-02 Patrick Juola: "In the case of any *particular* block cypher, with any *particular* key-space and any *particular* block size, &c, then the problem size is probably fixed (and P/NP is indeed a red herring). So proving that P == NP probably wouldn't affect the solution of DES much."
- 1998-11-03 Douglas A. Gwyn: "But the issue is not whether there is an *effective algorithm* for inverting *every* system of equations, which might bear on P?=NP. The statement was that proof of security of *any particular example* of a block cipher system would imply P=NP. That's what I doubt."
- 1998-11-02 Nicol So: "Whether a proof of security of a block cipher has anything to do with the question of P?=NP depends on how you formalize the notion of security."
- 1998-11-03 Nicol So: "...I meant to say...."
- 1998-11-03 Patrick Juola: "If I could prove that DES (or any particular sub-class of the general problem) *were* solvable in polynomial time, this would NOT prove that P == NP."
- 1998-11-03 bobs@rsa.com: "It is well known that problems exist that are HARDER than any problems in NP."
- 1998-11-03 Douglas A. Gwyn: "Please, check the attributions before posting."
- 1998-10-30 ssimpson@hertreg.ac.uk: "Are Schneier et al wrong?"
- 1998-10-30 Sandy Harris: "Methinks this argument is hopelessly flawed because the keylength in most ciphers cannot vary beyond a certain range & the whole P/NP distinction depends on reasoning for "in the limit" & "for sufficiently large N", so it cannot reasonably be applied."
- 1998-10-30 bobs@rsa.com: "Merely showing that breaking the key takes exponential time is NOT equivalent to proving it is NP-Complete."
- 1998-10-30 Patrick Juola: "Showing that breaking the key takes *provably* exponential time would suffice to show that P != NP."
- 1998-10-30 Paul Rubin: "If you can prove that *only* brute force works, the cipher is not in P."
- 1998-11-02 John Savard: "I think the idea is that while a _proof_ that only brute force works would indeed catapult cryptanalyzing it out of P, in general the fact that only brute force is known at present (which some people might take for a proof) certainly doesn't have anything to do with P versus NP."
- 1998-11-03 Shawn Willden: "Let me see if I can lay this out clearly and thoroughly enough that someone can point out the flaw in the reasoning...."
- 1998-11-02 John Savard: "Proving that brute force was not necessary would not prove P=NP...."
- 1998-10-31 Douglas A. Gwyn: "No, that's not even close to a proof...."
- 1998-11-02 Bryan G. Olson; CMSC (G): "Hmmm, I see it as kind of close."
- 1998-10-28 Bryan G. Olson; CMSC (G): "There are very few on this group who actually devote time and effort to looking into other peoples suggestions."
- 1998-10-21 dianelos@tecapro.com: "I would rather not use the word "break" to describe the successful cryptanalysis of a cipher." "...in the future Internet newsgroups will be the most important medium for communicating ideas while peer reviewed publications, as we know them today, will be less and less important."
- 1998-10-22 John Savard: "...with specific reference to the AES process, a cryptanalytic result that indicates a proposed cipher is less than _perfect_ is, quite properly, considered significant."
- 1998-10-23 W T Shaw: "...he said the describing a cipher in C would be OK with him, but not in a traditional *hardware* schematic."
- 1998-10-24 dianelos@tecapro.com: "What representation you choose is not a trivial matter. If a cipher designer always works sketching diagrams, in praxis he will artificially limit the range of ideas that he will consider."
- 1998-10-25 W T Shaw: "Having to work things out in solely by careful appearing and impressive sounding logic that may not be applicable to the real world is the essence of the scientific Greek Tragedy."
- 1998-10-26 Bruce Schneier: "I agree that 'break' is overused." "In a world where everyone is a publisher, editors become even more important."
- 1998-10-26 Mok-Kong Shen: "I think that the economy of description decides to some extent which way of presentation is to be prefered."
- 1998-10-26 Terry Ritter: "I recently posted a quote about this from the current IEEE Spectrum in another thread. Basically the idea is that the world is moving *away* from intermediaries who filter and decide for us, to the end-user (of clothes, of technical articles, etc.) surveying it all, and making the decision on what to select." "If math is a great advantage in understanding logic machines, why are logic machines not generally described that way? Why? Because schematics can be clearer, that's why."

- 1998-10-28 Frank O'Dwyer: "In a world where everyone can be a publisher, everyone can be an editor too."
- 1998-10-28 Patrick Juola: "Which implies that the value of good, worthwhile editing will continue to climb, just as the value of good *writing* has been climbing since the development of the Internet."
- 1998-10-28 Gurripato (x=nospam): "How would you then best describe Dobbertin´s attack on the compression function of MD5? Does it go all the way to demolition, plan brack, or just academic break?"
- 1998-10-27 Stefek Zaba: "'I Was A Chinese Sex Slave'"
- 1998-10-22 Mr. I. O. Yankle: "When I first read "Memo to the Amateur Cipher Designer" in Bruce Schneier's CRYPTO-GRAM, it was so clearly true and sensible to me that I expected it to gain immediate acceptance on sci.crypt and to even gain the status of 'required reading'. I still hope that this will be the case, but I can see now that it will take some time."
- 1998-10-22 W T Shaw: "...many of the thoughts have been expressed before."
- 1998-10-26 Terry Ritter: "I would hope that anyone reading Schneier's article would recognize that it is seriously flawed in many ways."
- 1998-10-27 Kery Minola: "You are really grasping at straws...."
- 1998-10-27 Xcott Craver: "!!! It's obvious that the memo did not mean 'prove' in the strict mathematical sense, but in the empirical sense."
- 1998-10-27 Douglas A. Gwyn: "The 'empirical proof' means very little since it can't allow for the eavesdropper's cryptanalytic abilities."
- 1998-10-27 Xcott Craver: "Are you suggesting that we should use something other than the scientific method?"
- 1998-10-27 W T Shaw: "To demand a single route to the truth is to prejudice against truths that may not be so conform to that path. This is the essence of what is wrong with what Bruce advocates, which is the same old tired argument we have heard for ages."
- t 98-10-02 Mike Zorn: "As an example, the benzene ring was not discovered by the 'scientific method'."
- 1998-10-28 Stefek Zaba: "Kekule's *intuition* about a possible structure for benzene may be implausible to explain as a deductive process...."
- 1998-10-28 Terry Ritter: "In normal science we innovate experiments to prove a result and get a new fact. In cryptography, we innovate experiments to prove a failure, and with a lack of failure we somehow leap to a conclusion of strength. This is a faulty leap. Crucially, the inability to break a cipher after much effort says nothing about its 'real' strength."
- 1998-10-28 dscott@networkusa.net: "ACtually if you come up with a good cipher you will not get it tested since they try to keep the rank of phony experts quite small."
- 1998-10-28 Douglas A. Gwyn: "...the so-called 'scientific method' is but one tool in our epistemological arsenal and ought not to be applied where it is ineffective."
- 1998-10-28 Xcott Craver: "Well, so what do you suggest as an alternative?"
- 1998-10-28 Bryan G. Olson; CMSC (G): "I agreed with Mr. Ritter on one point, but clearly Bruce got at least a 95%."
- 1998-10-26 Patrick Juola: "It's quite reasonable to use a person's ability to write clearly as a gauge for his/her ability to *think* clearly, given the observed high correlation between the two."
- 1998-10-27 Mok-Kong Shen: "In all fields of knowledge (including handcrafts) there are professionals and amateurs, the one group can't exist (by definition) without the other."
- 1998-10-27 Bruce Schneier: "While it is true that not every application need strong cryptography, this does not mean that these applications should look towards weak cryptography." "I think cryptography is one of the few branches of mathematics where the amateur can definitely compete with the professional."
- 1998-10-27 W T Shaw: "What you said above suggests the importance of diversity of method and manner which is opposed to the message of the Memo."
- 1998-10-27 W T Shaw: "Experience with a weaker version of an algorithm can teach you many things. If true scalable algorithms are involved, it remains the question of how strong do you want some implementation to be, always being able to make it infinitely stronger."
- 1998-10-28 Mok-Kong Shen: "The easier jobs have probably already all been discovered by the more capable professionals and done earlier, leaving the newcommers little chance. Thus I think the requirement of proving ones 'better' analysis capability is suppressive for novel design ideas from coming up."
- 1998-10-28 Bruce Schneier: "...people who have not demonstrated their ability to break algorithms are unlikely to develop algorithms that cannot easily be broken. I don't believe the easier jobs havae all been taken."
- 1998-10-28 Mok-Kong Shen: "These are so to say 'ready foods' for the would-be professionals on the way to their true professional status. Why have these been so rarely attacked? Or are there barely any would-be professionals around perhaps?"
- 1998-10-28 Bruce Schneier: "Because people are busy. Because not everyone has time to spend weeks (or days or even hours) analyzing every random cipher that comes across their desk. Because the designs are not pubished, so the breaks are not publishable."
- 1998-10-29 Mok-Kong Shen: "I disagree. The would-be professionals are busy in attempting to proving their 'better'... analyis capability through cracking algorithms that are presumably hard."
- 1998-10-29 Bruce Schneier: "Many of us have breaks of amateur ciphers, ones that appear on sc.crypt, get patents, or are used opterationally, that we just don't have time to write up or flesh out. It's just not worth the bother."
- 1998-10-29 Mok-Kong Shen: "You said that because people are busy no one has the time to look at the amateur ciphers that are unpublished, etc. etc. I argued, hopefully convincingly and clearly, that at least Terry Ritter's designs do deserve being analyzed...."
- 1998-10-29 Bruce Schneier: "You know, I don't want to pick on Ritter in particular here. I don't know about whether his designs "deserve" to be analyzed; that is a value judgment. I don't know if they are strong or weak."
- 1998-10-30 dscott@networkusa.net: "...I guess I sometimes do agree with limited parts of what Bruce Babels out."
- 1998-10-30 Mok-Kong Shen: "Very sorry that I am not on your side. Quite a lot of what Bruce Schneier said does correspond to the reality (a sad reality though)...."
- 1998-10-30 : "I'd like to see a group that tries to develop and to break amateur ciphers - not as a group of cryptographers that develope strong ciphers, but as cryptanalyticers (something like the ACA but working with computers and modern cryptanalysis)."
- 1998-10-30 W T Shaw: "Many in the ACA are working with computers and extending their capabilities. The first hurdle has been in developing automated means of solving all ciphers in the ACA stable."
- 1998-10-31 Douglas A. Gwyn: "Actually, the ACA does have a section devoted to computers. But it needs more members!"
- 1998-10-30 Mok-Kong Shen: "Are you saying that the academia neglects the patent publications?"
- 1998-10-30 dscott@networkusa.net: "I guess then you greatly underestamate the EGO of the phony crypto gods."
- 1998-10-30 Bruce Schneier: "Ritter is an example of someone who does not publish (in an academic sense) but does patent. His writings are generally ignored by the academic community. I'm sorry this is true; I'd like it to be different."
- 1998-10-30 Mok-Kong Shen: "Patents cannot be ignored by the academic community. If one develops a new cipher, he needs to know whether he doesn't infringe on someone's patents."
- 1998-10-30 Bruce Schneier: "Patents are not considered peer-reviewed publications in academia."
- 1998-11-02 Mok-Kong Shen: "In A. J. Menezes et al. a whole chapter, Chap. 15, is devoted to 'Patents and Standards'. There they write: 'This chapter discusses two topics which have significant impact on the use of cryptology in practice: patents and standards.'"
- 1998-11-02 Patrick Juola: "The overall standing of patent review is sufficiently low that the people who have the authority to decide what does and doesn't constitute 'peer review' have decided that patents don't cut it."
- 1998-11-02 JPeschel: "Why not go here and look around...."
- 1998-11-03 Mok-Kong Shen: "There are scientific conferences where the majority of the program committe, even including the chair, are not from universities. Are those who are not academics not 'peers' doing the review and are not equivalent to those who have university positions in the process?"
- 1998-11-06 Bruce Schneier: "I believe you would be amazed by what gets through the patent office. The only thing they regularly catch are perpetual motion machines...."
- 1998-11-06 Bruce Schneier: "Oops. I meant 'community.'"
- 1998-11-10 Mok-Kong Shen: "My sincere apology all to readers of the group for having wasted bandwidth."
- 1998-11-11 Mok-Kong Shen: "If the academics choose to ignore the patent publications and claim that only the papers in journals edited by them are scientific contributions (I doubt this), then they are not practicizing science but 'religion'!"
- 1998-11-11 Stefan Axelsson: "...the majority of researchers recognise that there are difficult, and fundamental problems with referring to URL:s, or other forms of transient communication."
- 1998-11-11 W T Shaw: "Scientific truth is what is valuable, preferable to that of a paper published through a process that might ignore aspects by limiting debate of the particulars."
- 1998-11-12 Stefan Axelsson: "Now, of course there are references, and references, but if one resorts to to building ones argument on a reference that the reader cannot himself verify, then of course one must question why...."

- 1998-11-12 Joseph K. Nilaad: "I can see that URL may be short life, but as long as it lives, it should be considered valid reference."
- 1998-11-12 Patrick Juola: "But http://www.bedrock.com/~fleems isn't nearly as helpful if the domain no longer exists and I can't even tell who did the work to phone him."
- 1998-11-18 Joseph K. Nilaad: "Likewise, what if the publishers like Random house no longer exist. So what If referenced URL no longer exist. At least you're being *honest* about it."
- 1998-11-18 Patrick Juola: "The basic problem is that telling him that it's at http:[...]/~fleems is borderline useless."
- 1998-11-23 Coen L.S. Visser: "The problem is more serious than just a disappearing URL. What if the URL still exists, but the content has changed."
- 1998-11-23 Arnoud "Galactus" Engelfriet: "How about downloading the relevant documents from the Web and putting them on a CD-ROM, which is distributed together with the report?"
- 1998-11-23 Coen L.S. Visser: "That would be really nice of course but as you already state it has a lot of practical problems."
- 1998-11-23 Stefan Axelsson: "What is needed, is some other, resilient, long lasting, redundant third party storage of references, such as a library is for printed material today."
- 1998-11-23 Coen L.S. Visser: "Libraries would be ideal for that task...."
- 1998-11-14 Stefan Axelsson: "...the average time from research to publication in a refereed journal today is two years. Many/most of those URL:s will be dead by the time the paper leaves the presses."
- 1998-11-16 Joseph K. Nilaad: "By the time it is published, the matterial may not be applicable 2-3 years later." "...just because publishing via URL is relatively short life comparing with hard copies, it doesn't mean we should not give publishers their credits. Unless, if you think that swindling someone's idea is OK."
- 1998-11-17 W T Shaw: "If you can give credit, fine."
- 1998-11-17 Stefan Axelsson: "...If your only motivation for including a reference is to acknowledge someone else's, idea, then the name of said person would (in general) do nicely. If you include a URL, it should be with the knowledge that it is/will become useless to the reader in a very short period of time."
- 1998-11-12 Terry Ritter: "This addresses the *convenience* of Science to the reader. But it ignores the *responsibility* of the author and the *requirement* of scientific publication to acknowledge the previous work, the source of the inspiration (rarely is any work completely original). If that previous work came in a private letter, so be it."
- 1998-11-14 Stefan Axelsson: "...if you *build* your argument on something you reference, then this reference should be reliably available to your peers."
- 1998-11-11 Bruce Schneier: "...I consider myself avant guard by citing email messages, URLs, and patents in my papers. I take some flak for it, but I do it anyway. Most others don't bother...."
- 1998-11-11 Mok-Kong Shen: "To be an avantguard is one thing yet not to mention a relevant fact that a scientist himself is MOST familiar...."
- 1998-11-11 Bruce Schneier: "Yeah. Sure. You're right. Whatever."
- 1998-11-11 W T Shaw: "You have that in common with Ritter, as I recall."
- 1998-11-11 Bruce Schneier: "There are others, too. Currently the academic community is still trying to figure out how to handle URL references."
- 1998-11-12 W T Shaw: "Fair use should mean that you could post the reference if it disappeared. Important things change from what is printed in journals and books too, job titles, mailing addresses and phone numbers. Actual technical mistakes are rather hard to reverse as well in fixed media; note the increased leaning on the web for current updates."
- 1998-11-12 Bruce Schneier: "The page I look at when I write my paper may or may not be the same page you look at when you check my reference."
- 1998-11-12 Tim Bass: "With the current state of the network, it is quite unprofessional to reference URLS."
- 1998-11-12 Terry Ritter: "...I often find myself dealing with an different version of a well-known work than someone else is quoting. I handle this, when necessary, by going to the library and getting 'the' reference."
- 1998-11-13 Mok-Kong Shen: "Informations on the internet, in particular Web, is getting archived."
- 1998-11-13 Patrick Juola: "...assuming the average document half-life is about six months (which I pulled out of thin air, but seems about right), then you'll need to buy the entire Web in terms of disk capacity EVERY YEAR."
- 1998-11-18 Coen L.S. Visser: "I think the author making the reference should be responsible for archiving the particular web page in case the original reference becomes invalid."
- 1998-11-18 Patrick Juola: "I think that's about the fourth most unreasonable assertion I've heard in my life."
- 1998-11-23 Coen L.S. Visser: "...if you write books for a living and you have a web page, I believe the chances are quite high that your (new) web page can be found."
- 1998-11-13 Joseph K. Nilaad: "This is just a thought of handling referred URL documents: If a document has references from any URL, those URL referrences must be electronically signed."
- 1998-11-17 Mok-Kong Shen: "...paper publications actually also have the same problem."
- 1998-11-17 Stefan Axelsson: "...if the IEEE for example where to say, OK, to h*ll with the dead trees, let there be business as usual, but on the web instead, then of course, (almost) all that which is the IEEE would transfer to the electronic medium, and little would have to change." "The situation with everyone "publishing" their material is so far removed from this that I don't know where to start."
- 1998-11-17 Mok-Kong Shen: "How do we know that this document is really from the person having the name X?"
- 1998-11-17 Patrick Juola: "The IEEE (for example) has implicitly 'signed' or 'authenticated' the claims made in its published work."
- 1998-11-23 Stefan Axelsson: "...the beauty of there being several hard copies made of each publication, makes it trivial for the reader to get his material from several sources, should he lack trust in any single one of them."
- 1998-11-17 W T Shaw: "You could mail or email a question to the author."
- 1998-11-17 Patrick Juola: "...even if you send mail and get a response back, how do you know that you're getting the right person?"
- 1998-11-22 W T Shaw: "There is something said for meeting people physically, creating a history on which verification can be based."
- 1998-11-11 Bruce Schneier: "Almost all patents are examined by almost nobody."
- 1998-11-11 Mok-Kong Shen: "...large chemical firms need people knowledgeable in such patents in order that they can do their business properly."
- 1998-11-11 Bruce Schneier: "I'm going to drop the thread."
- 1998-11-12 Mok-Kong Shen: "That patents are important in the 'practice' (as against pure theory) of a large number of professions should be well-known."
- 1998-11-13 Mok-Kong Shen: "...only three days after the issue of the patent it is already to be found on a Web page...."
- 1998-11-18 Mok-Kong Shen: "Patents play in science and technology a significant role which cannot be and is not ignored by the academic community...."
- 1998-11-19 Mok-Kong Shen: "I was only arguing about the VALUE of patent documents which Bruce Schneier negated, saying that these are not even publications."
- 1998-11-12 Joseph K. Nilaad: "...why should one work for someone for free? The patents are owned by somebody!"
- 1998-11-16 Denning Langston: "Processes that create useful chemical compounds efficiently or cheaply are patentable, and specific uses of chemical compounds are patentable (pharmaceuticals, pesticides, herbicides, etc.), but chemical compounds in and of themselves are not patentable."
- 1998-11-16 Mok-Kong Shen: "...patents... contain... essential and valuable scientific informations which should not be ignored by the academics (those at the universities and the academies of sciences) and, as far as I can make out, are indeed not largely ignored by them (the converse was argued by Bruce Schneier.)"
- 1998-11-10 Joseph K. Nilaad: "My point is that it doesn't matter whether it is amateur or expert who design the crypto, patent or not, we all want the best crypto possible. If AES confine to non patent algorithm, I think it is very narrow minded."
- 1998-11-10 Patrick Juola: "No apologies for patent agents should be necessary -- they're overworked civil servants doing the best they can under adverse conditions. But they're certainly not the peers of the authors of papers at CRYPTO'97."
- 1998-11-10 : "Without patents all new inventions would have to be kept secret to keep others from copying it." "But it is really not their job to test an encryption algorithm for strength."
- 1998-11-10 Andrew Haley: "If the AES is to be universally used, it must not be encumbered by royalties."
- 1998-11-11 W T Shaw: "AES has several implications, only one of them be that could replace lots of others."
- 1998-11-10 Bruce Schneier: "I have nothing against patented algorithms. People are welcome to patent algorithms. I see no reason to implement patented algorithms when there are unpatented alternatifves. This is just good economics. I see no reason to perform free analysis on patented algorithms unless there is a good reason to do so."
- 1998-11-11 malinov@mindless.com: "Last I checked, cryptography is examined at the USPTO in art unit 2766 by three primary examiners, two juniors on the verge of becoming primaries and four juniors still in their first six months."
- 1998-11-12 Terry Ritter: "This is *temporary* economics. By failing to compensate the work actually performed, we fail to build a profit-based business of cipher design. We have ciphers, yes. But we do not have a continuing business of cipher design, along with the expensive expertise and corporate history associated with other technologies."
- 1998-11-13 : "Free software has a long tradition and no other software is developed faster and more continuous than free software."

- 1998-11-13 Andrew Haley: "A successful AES candidate must be universal. This means that it must be used everywhere, in both free and unfree software. A patented algorithm may not be used in free software, so cannot be used universally."
- 1998-11-13 Bryan G. Olson; CMSC (G): "If Bob works on an algorithm for free, then if he finds a weakness he gets to publish, and if he doesn't he never has reveal he tried."
- 1998-11-13 Bruce Schneier: "Occasionally a company hires us to review an open cryptographic primitive, and allows us to publish our results."
- 1998-11-16 John Savard: "...the absence of a thriving cryptography industry means that the ciphers available for use are not as strong, or as well-analyzed, as they might be."
- 1998-11-11 Joseph K. Nilaad: "This really bothers me."
- 1998-10-30 Patrick Juola: "The existence of embarassingly large numbers of thoroughly ludicrous patents is well-documented."
- 1998-11-04 W T Shaw: "Cryptography is a special class...."
- 1998-11-05 dscott@networkusa.net: "If the NSA is doing its job at all any encryption that is used at all on the net would be analyzed by them."
- 1998-11-04 Andrew Carol: "In my military comm days, we had a custom IO proccessor which could handle word sizes from 5 upto 32 bits, in either big or little endian, in either positive or negative logic, with any parity (or even multiple parity per word)."
- 1998-11-05 dscott@networkusa.net: "...I still feel that having a non mulitple of 8 makes it even safer."
- 1998-11-05 Andrew Carol: "One's complement is, in my humble opinion, a real waste."
- 1998-11-05 dscott@networkusa.net: "At least the 1's complement had the same range of numbers in the positive and negative direction...."
- 1998-11-05 Andrew Carol: "...I end up using features of 2's complement almost everyday, and can't think of the last time I wished I was using the 1's complement"
- 1998-11-5 R H Braddam: "Your post reminds me of the AN/FST-2B data processor."
- 1998-11-05 Andrew Carol: "I worked on the follow-on system, FYQ-9?"
- 1998-11-06 Douglas A. Gwyn: "The right to privacy is not explicitly spelled out...."
- 1998-11-06 David Sternlight: "...the government may compel productions of one's private papers via lawful subpoena despite the Constitutional 'right to be secure in one's papers'."
- 1998-11-11 Scott Nelson: "The existence of the general right to privacy has never seriously been questioned."
- 1998-11-12 pstromer@my-dejanews.com: "'We recently referred [p*485] in Mapp v. Ohio, 367 U.S. 643, 656, to the Fourth Amendment as creating a 'right to privacy, no less important than any other right carefully an particularly reserved to the people.'"
- 1998-11-12 R H Braddam: "I hope this ends the discussion about the right to privacy -- whether it exists or not. It does, not just in my opinion, but in the opinion of the Congress and the Supreme Court."
- 1998-11-18 lamontg@bite.me.spammers: "It has never, however, been an explicit part of the Constitition."
- 1998-11-06 Bryan G. Olson; CMSC (G): "Two's complement is simply arithmetic mod 2^WordSize, while one's complement is mod (2^WordSize)-1."
- 1998-11-07 : "...it is true that less circuitry is required to add a negative integer to a positive integer in two's complement...."
- 1998-11-08 Douglas A. Gwyn: "I've programmed both, and either representation is reasonable for most purposes."
- 1998-11-04 W T Shaw: "I expect they already have looked at your various algorithms. After all it is their job to do this sort of thing."
- 1998-11-06 Bruce Schneier: "The NSA does not comment on patentability. Actually, I'm pretty sure they don't review crypto patent apps anymore."
- 1998-11-06 W T Shaw: "I don't expect NSA to be incompetent, which means it should it sops up and funnels ALL the easy leads to the company store for analysis."
- 1998-11-10 Terry Ritter: "It is my understanding that there is "a NSA desk" in the PTO which does review crypto applications. If so, it must be a tough job." "My Dynamic Substitution patent file actually *disappeared* from the PTO for a while in 1990."
- 1998-11-10 malinov@mindless.com: "Applications by US citizens which involve real cryptography are copied; the copy is sent to someone at NSA for review."
- 1998-11-10 Terry Ritter: "I was eventually bucked up to the department head Herself and she wouldn't tell me where the file was or why, which seems unusual to this day."
- 1998-11-11 malinov@mindless.com: "Secrecy orders are only imposed on government owned systems, usually classified from birth."
- 1998-11-10 John Savard: "...unbreakable encryption is already a reality for anyone who wants it."
- 1998-11-10 Bo Dömstedt: "What would happen if some foreigner, such as me, would file a cipher patent application?"
- 1998-11-11 malinov@mindless.com: "If you filed an application from Sweden in the US, it would not even be shown to NSA."
- 1998-10-30 Terry Ritter: "I have no doubt that understanding the patent literature is substantially more difficult than understanding academic papers. Still, it *is* published technology. Failing to know published technology means failing to be an expert in the field." "It's *not* true that I don't publish; I just don't publish in a particular form and outlet."
- 1998-10-30 Bruce Schneier: "Perhaps it is true that someone who does not regularly read the patent output stream should not be considered an expert in the field. I don't know. None of this is related to my point, which is about what actually does happen."
- 1998-10-31 dscott@networkusa.net: "How political correct."
- 1998-10-31 JPeschel: "Look, Dave, you aren't helping your cause...."
- 1998-10-31 Remo A. Linky: "Thanks for hanging in there!"
- 1998-10-31 Bruce Schneier: "Thanks."
- 1998-10-31 dscott@networkusa.net: "...even though my rules are unfair to bad for you other guys and I don't mind telling about these unfair rules of mine."
- 1998-10-31 W T Shaw: "...please don't consider our academic and philosophical criticisms as personal attacks; heated debate can merely mean that people are getting down to the most important aspects of their differences...."
- 1998-10-31 Bruce Schneier: "I like the academic and philosophical criticisms; that's why I stay here."
- 1998-10-31 W T Shaw: "Sometimes it is best to see how far you can go on your own without tainting your thoughts with the misgivings of others."
- 1998-10-30 John Savard: "It should be noted, though, that Terry Ritter has had academic publications in the past; three or so papers in Cryptologia on Dynamic Substitution and related topics."
- 1998-10-30 Bruce Schneier: "Thanks for pointing that out. I had meant to, but forgot."
- 1998-10-31 dscott@networkusa.net: "Get real your iragance is showing through."
- 1998-10-31 dscott@networkusa.net: "John Bruce most likely was in his usual put people down mode...."
- 1998-10-31 Terry Ritter: "Let's just try to hold it down a little."
- 1998-10-28 Terry Ritter: "...my stuff is *scalable*: We have *no* *doubt* that the ultimate scaled-down versions will be weak. By using few component types and an overall regular structure, I hope to *expose* every avenue into the design."
- 1998-10-28 Bruce Schneier: "It would be better if you would put up strawman designs, so that people would have something concrete to analyze...."
- 1998-10-30 John Savard: "...the quote as I saw it in later posts, out of context, seems to say that here are designs that even amateurs can break."
- 1998-10-28 Douglas A. Gwyn: "...occasionally breakthroughs in knowledge are made by 'dabblers' who work outside the mainstream. However, the odds are still that in any given case a dabbler is wasting his time."
- 1998-10-29 W T Shaw: "I suppose I am an optomist, but fairness seems to mean giving someone the benefit of the doubt, whether you want to or not."
- 1998-10-29 Bruce Schneier: "If someone tries hard enough, he will wear his 'benefit of the doubt' out."
- 1998-10-30 dscott@networkusa.net: "Not sure if I should give you the benefit of the doubt since you only tend to look down on the masses and prevent any one with a good idea to get a fair chance."
- 1998-10-31 fungus: "A serious analysis of something like ScottXX is a serious undertaking, probably a couple of months of hard work."
- 1998-11-09 newWebsite: "The website is ready for you!"

```
Subject: Memo to the Amateur Cipher Designer
Date: Sat, 17 Oct 1998 23:35:28 GMT
```

From: schneier@counterpane.com (Bruce Schneier)
Message-ID: <36292906.1151332@news.visi.com>
Newsgroups: sci.crypt
Lines: 152

This was in the October CRYPTO-GRAM, but I thought I'd run it through sci.crypt, since so many people seem to be asking questions on the topic.

Bruce

Memo to the Amateur Cipher Designer

Congratulations. You've just invented this great new cipher, and you want to do something with it. You're new in the field; no one's heard of you, and you don't have any credentials as a cryptanalyst. You want to get well-known cryptographers to look at your work. What can you do?

Unfortunately, you have a tough road ahead of you. I see about two new cipher designs from amateur cryptographers every week. The odds of any of these ciphers being secure are slim. The odds of any of them being both secure and efficient are negligible. The odds of any of them being worth actual money are virtually non-existent.

Anyone, from the most clueless amateur to the best cryptographer, can create an algorithm that he himself can't break. It's not even hard. What is hard is creating an algorithm that no one else can break, even after years of analysis. And the only way to prove that is to subject the algorithm to years of analysis by the best cryptographers around.

"The best cryptographers around" break a lot of ciphers. The academic literature is littered with the carcasses of ciphers broken by their analyses. But they're a busy bunch; they don't have time to break everything. How do they decide what to look at?

Ideally, cryptographers should only look at ciphers that have a reasonable chance of being secure. And since anyone can create a cipher that he believes to be secure, this means that cryptographers should only look at ciphers created by people whose opinions are worth something. No one is impressed if a random person creates an cipher he can't break; but if one of the world's best cryptographers creates an cipher he can't break, now that's worth looking at.

The real world isn't that tidy. Cryptographers look at algorithms that are either interesting or are likely to yield publishable results. This means that they are going to look at algorithms by respected cryptographers, algorithms fielded in large public systems (e.g., cellular phones, pay-TV decoders, Microsoft products), and algorithms that are published in the academic literature. Algorithms posted to Internet newsgroups by unknowns won't get a second glance. Neither will patented but unpublished algorithms, or proprietary algorithms embedded in obscure products.

It's hard to get a cryptographic algorithm published. Most conferences and workshops won't accept designs from unknowns and without extensive analysis. This may seem unfair: unknowns can't get their ciphers published because they are unknowns, and hence no one will ever see their work. In reality, if the only "work" someone ever does is in design, then it's probably not worth publishing. Unknowns can become knowns by publishing cryptanalyses of existing ciphers; most conferences accept these papers.

When I started writing _Applied Cryptography_, I heard the maxim that the only good algorithm designers were people who spent years analyzing existing designs. The maxim made sense, and I believed it. Over the years, as I spend more time doing design and analysis, the truth of the maxim has gotten stronger and stronger. My work on the Twofish design has made me believe this even more strongly. The cipher's strength is not in its design; anyone could design something like that. The strength is in its analysis. We spent over 1000 man-hours analyzing Twofish, breaking simplified versions and variants, and studying modifications. And we could not have done that analysis, nor would we have had any confidence in that analysis, had not the entire design team had experience breaking many other algorithm designs.

A cryptographer friend tells the story of an amateur who kept bothering him with the cipher he invented. The cryptographer would break the cipher, the amateur would make a change to "fix" it, and the cryptographer would break it again. This exchange went on a few times until the cryptographer became fed up. When the amateur visited him to hear what the cryptographer thought, the cryptographer put three envelopes face down on the table. "In each of these envelopes is an attack against your cipher. Take one and read it. Don't come back until you've discovered the other two attacks." The amateur was never heard from again.

I don't mean to be completely negative. People occasionally design strong ciphers. Amateur cryptographers even design strong ciphers. But if you are not known to the cryptographic community, and you expect other cryptographers to look at your work, you have to do

several things:

1. Describe your cipher using standard notation. This doesn't mean C code. There is established terminology in the literature. Learn it and use it; no one will learn your specialized terminology.

2. Compare your cipher with other designs. Most likely, it will use some ideas that have been used before. Reference them. This will make it easier for others to understand your work, and shows that you understand the literature.

3. Show why your cipher is immune against each of the major attacks known in literature. It is not good enough just to say that it is secure, you have to show why it is secure against these attacks. This requires, of course, that you not only have read the literature, but also understand it. Expect this process to take months, and result in a large heavily mathematical document. And remember, statistical tests are not very meaningful.

4. Explain why your cipher is better than existing alternatives. It makes no sense to look at something new unless it has clear advantages over the old stuff. Is it faster on Pentiums? Smaller in hardware? What? I have frequently said that, given enough rounds, pretty much anything is secure. Your design needs to have significant performance advantages. And "it can't be broken" is not an advantage; it's a prerequisite.

5. Publish the cipher. Experience shows that ciphers that are not published are most often very weak. Keeping the cipher secret does not improve the security once the cipher is widely used, so if your cipher has to be kept secret to be secure, it is useless anyway.

6. Don't patent the cipher. You can't make money selling a cipher. There are just too many good free ones. Everyone who submitted a cipher to the AES is willing to just give it away; many of the submissions are already in the public domain. If you patent your design, everyone will just use something else. And no one will analyze it for you (unless you pay them); why should they work for you for free?

7. Be patient. There are a lot of algorithms to look at right now. The AES competition has given cryptographers 15 new designs to analyze, and we have to pick a winner by Spring 2000. Any good cryptographer with spare time is poking at those designs.

If you want to design algorithms, start by breaking the ones out there. Practice by breaking algorithms that have already been broken (without peeking at the answers). Break something no one else has broken. Break another. Get your breaks published. When you have established yourself as someone who can break algorithms, then you can start designing new algorithms. Before then, no one will take you seriously.

Creating a cipher is easy. Analyzing it is hard.

See "Self-Study Course in Block Cipher Cryptanalysis":
http://www.counterpane.com/self-study.html
****
Bruce Schneier, President, Counterpane Systems Phone: 612-823-1098
101 E Minnehaha Parkway, Minneapolis, MN 55419 Fax: 612-823-1590
Free crypto newsletter. See: http://www.counterpane.com

Subject: Re: Memo to the Amateur Cipher Designer
Date: Sun, 18 Oct 1998 04:20:15 GMT
From: george.barwood@dial.pipex.com (George Barwood)
Message-ID: <362967c9.4415110@news.dial.pipex.com>
References: <36292906.1151332@news.visi.com>
Newsgroups: sci.crypt
Lines: 14

On Sat, 17 Oct 1998 23:35:28 GMT, schneier@counterpane.com (Bruce Schneier) wrote in part:

> Algorithms posted to Internet newsgroups by unknowns won't get a second glance.

I disagree - some time ago I posted an algorithm to sci.crypt, and recieved a quick (and useful) analysis from David Wagner. The algorithm was not strong against known-plaintext attack, but this was as expected (the design aim was speed at all costs).

Not that I disagree with the intent or conclusions of your article - but I don't this statement holds up.

George

Subject: Re: Memo to the Amateur Cipher Designer
Date: 18 Oct 1998 06:07:01 -0700
From: Karl-Friedrich Lenz
Message-ID: <70cp5l$jbu@edrn.newsguy.com>
References: <362967c9.4415110@news.dial.pipex.com>
Newsgroups: sci.crypt

Lines: 22

In article , george.barwood@dial.pipex.com says...
>
>On Sat, 17 Oct 1998 23:35:28 GMT, schneier@counterpane.com (Bruce
>Schneier) wrote in part:
>
>>Algorithms posted to Internet newsgroups by unknowns won't get a second glance.
>
>I disagree - some time ago I posted an algorithm to sci.crypt, and
>recieved a quick (and useful) analysis from David Wagner. The
>algorithm was not strong against known-plaintext attack, but this was
>as expected (the design aim was speed at all costs).
>
>Not that I disagree with the intent or conclusions of your article -
>but I don't this statement holds up.

Probably Mr. Schneier intended to say "not a second glance by professionals in scientific papers", which might be true. But the level of sci.crypt is not that low, and there seem to be quite a lot of people ready to have a swing at new ideas.

Karl-Friedrich Lenz :-)
www.toptext.com/crypto

Subject: Re: Memo to the Amateur Cipher Designer
Date: Sun, 18 Oct 1998 15:00:36 GMT
From: schneier@counterpane.com (Bruce Schneier)
Message-ID: <362a0287.3103532@news.visi.com>
References: <362967c9.4415110@news.dial.pipex.com>
Newsgroups: sci.crypt
Lines: 23

On Sun, 18 Oct 1998 04:20:15 GMT, george.barwood@dial.pipex.com
(George Barwood) wrote:

>On Sat, 17 Oct 1998 23:35:28 GMT, schneier@counterpane.com (Bruce
>Schneier) wrote in part:
>
>> Algorithms posted to Internet newsgroups by unknowns won't get a second glance.
>
>I disagree - some time ago I posted an algorithm to sci.crypt, and
>recieved a quick (and useful) analysis from David Wagner. The
>algorithm was not strong against known-plaintext attack, but this was
>as expected (the design aim was speed at all costs).
>
>Not that I disagree with the intent or conclusions of your article -
>but I don't this statement holds up.

You're right. There are exceptions to this. Agreed.

Bruce
****
Bruce Schneier, President, Counterpane Systems Phone: 612-823-1098
101 E Minnehaha Parkway, Minneapolis, MN 55419 Fax: 612-823-1590
Free crypto newsletter. See: http://www.counterpane.com

Subject: Re: Memo to the Amateur Cipher Designer
Date: 18 Oct 1998 17:17:12 +0200
From: Jon Haugsand <haugsand@procyon.nr.no>
Message-ID: <yzobtn9nblz.fsf@procyon.nr.no>
References: <362a0287.3103532@news.visi.com>
Newsgroups: sci.crypt
Lines: 19

* Bruce Schneier
| >> Algorithms posted to Internet newsgroups by unknowns won't get a second glance.
| >
| >I disagree - some time ago I posted an algorithm to sci.crypt, and
| >recieved a quick (and useful) analysis from David Wagner. The
| >algorithm was not strong against known-plaintext attack, but this was
| >as expected (the design aim was speed at all costs).
|
| You're right. There are exceptions to this. Agreed.

Actually, wouldn't this be a good way to train oneself with
cryptoanalyzing? Breaking amateur ciphers posted to the usenet?

--
Jon Haugsand
Norwegian Computing Center, <http://www.nr.no/engelsk/>
<mailto:haugsand@nr.no> Pho: +47 22852608 / +47 22852500,
Fax: +47 22697660, Pb 114 Blindern, N-0314 OSLO, Norway

Subject: Re: Memo to the Amateur Cipher Designer
Date: Mon, 19 Oct 1998 04:09:14 GMT
From: schneier@counterpane.com (Bruce Schneier)
Message-ID: <362abb52.2020632@news.visi.com>
References: <yzobtn9nblz.fsf@procyon.nr.no>

Newsgroups: sci.crypt
Lines: 25

On 18 Oct 1998 17:17:12 +0200, Jon Haugsand <haugsand@procyon.nr.no>
wrote:

>* Bruce Schneier
>| >> Algorithms posted to Internet newsgroups by unknowns won't get a second glance.
>| >
>| >I disagree - some time ago I posted an algorithm to sci.crypt, and
>| >recieved a quick (and useful) analysis from David Wagner. The
>| >algorithm was not strong against known-plaintext attack, but this was
>| >as expected (the design aim was speed at all costs).
>|
>| You're right. There are exceptions to this. Agreed.
>
>Actually, wouldn't this be a good way to train oneself with
>cryptoanalyzing? Breaking amateur ciphers posted to the usenet?

Definitely. I think it's the best way. Not only do you get
experience breaking ciphers, but you get some very easy ones to start
on.

Bruce
****
Bruce Schneier, President, Counterpane Systems Phone: 612-823-1098
101 E Minnehaha Parkway, Minneapolis, MN 55419 Fax: 612-823-1590
Free crypto newsletter. See: http://www.counterpane.com

Subject: Re: Memo to the Amateur Cipher Designer
Date: Sat, 17 Oct 1998 22:33:44 -0600
From: jgfunj@EnqvbSerrGrknf.pbz (W T Shaw)
Message-ID: <jgfunj-1710982234000001@dialup175.itexas.net>
References: <36292906.1151332@news.visi.com>
Newsgroups: sci.crypt
Lines: 110

In article <36292906.1151332@news.visi.com>, schneier@counterpane.com
(Bruce Schneier) wrote:

> This was in the October CRYPTO-GRAM, but I thought I'd run it through
> sci.crypt, since so many people seem to be asking questions on the
> topic.
>
> Bruce.
....

There have been many such discussions which marry some good advice with propaganda, serving the status quo rather than being inclusive of all attempts at improvement in the condition of man. A contrived obstacle course means being sure that few can finish, and more are discouraged from even trying. Those that do run the gauntlet and break the tape seem to confirm its validity to the blinded faithful, not withstanding the best intentions of those who would sit in judgement, doing the best they can to feel that the whole process is of inordinate value.

As with any presentation, you are encouraged to find weaknesses in what is included in the prior posting in this thread. Authoritarianism is always subject to incompleteness in information that conflicts with its adopted views; and, the stronger it is the more vocal it is in denouncing whatever differs with it. Intolerance ain't pretty.

Since sound reasoning is essential in cryptography: If you know where your feet are, you should be able to cut through the nonsense to glean something even useful from the talk. Much of the content is not new at all, but contrived decades ago, and seeks to hamstring the possibilities of the present to the hinderances of the past, more especially in this subject of ours, and not further the open art at all. The scripting of the elements is in the form of an arrangement in supportative order for argument's sake so they sound more reasonable that they are. The caveats do form comfortable enclaves for those that want to excuse the rest of the stuff.

Remember, the only excuse for formal education is learning how to learn. The end ideal is to become a self-starter in your search for truth, not requiring so many hours credit in order to have particular ability. What is to be acquired is being able to DO rather than always having to ask permission and direction for your occupations. When this honest goal of finding your own direction is realized, it means that you are weaned. It means that you are no longer required to seak an academic teat, or kiss customary areas of despoiled anatomy.

You still have the right to seek helpful advice for its own sake, but no obigation to bow and scrape for the priviledge. Good information is not to be cloistered.

You are allowed to judge legitimacy on intrinsic content rather than whether it contradicts prior cannonized scripture. You are encouraged in true scientific tradition to test and inquire into the nature of anything that has been spread before as the gospel.

If you are overly addicted to the opinons of certain people, you tend to acquire their prejudices; afterwards, know that discovering any flaws is

prohibited, and severly punished by excommunication, which has always been a religious act aimed at the unfaithful so as to humiliate and silence them. This technique is often used as well against those that do not buy the bit up front.

So often those that tout a regimen are just saying that it worked for them, so it can do the same for you. You can eat the blood pudding of tradition as long as you like, or you can graduate in informal elegancy, freedom of thought being its own reward.

If you are not ready to fly, you may crash, which is preferable to being stoned or shot down as a heretic in the other model. You then have the option to dust yourself off, learn from your mistakes, and flap your wings again.

Reinforcing the status quo means going nowhere not on the approved map; innovation and creativity mean taking new and unorthodox approaches, and sometimes finding that assumed ground rules are merely generalizations that are not always true.

Life is far more variable than anyone can realize. It is such that you can almost have nothing on the surface in common with whole groups of people. This means that methods that work for some are going to be rejected as bad style by others. The challenge is not to forcefully remake everyone else in your own image, but to realize that noone has a lock on the path to truth. It should be self-evident that what leads you is the greater good rather than finding a way to get more articles published than someone else.

In crypto, as in many other fields, sufficient study will lead you to agreement with lots of what passes for acceptable thought. It can allow you to unmask areas that have been glossed over. I would never discourage someone from going it alone in a quest; so much in science is the product of the dedicated contrarians who focused on a star that others wanted to excuse as an photographic artifact. Be constrained only by those barriers you show to be actually there. Cryptography is still wide open to new concepts, as well as novel unifying ideas that put older methods in prospective.

Bruce is a good soldier, but some don't march to the same drummer. I would like to believe that anyone as intelligent as he appears to be would serve less in the role of retelling so many false echos from the past. He continually tells us how difficult good cryptography is; I suppose that reflects his experience. I am sure that he would like to make it easier for others to learn what he has without going down the same path, yet he would recommend it still.

Yet, I would not discourage him either from any cryptological endeavor, as I would not do that to anyone.
--

Insanity means doing the same thing over and over again and expecting different results...like CDA2.

Decrypt with ROT13 to get correct email address.

User-Agent: tin/pre-1.4-980618 (UNIX) (AIX/4-1)
Cache-Post-Path: server.cuug.ab.ca!unknown@ibm.cuug.ab.ca

Subject: Re: Memo to the Amateur Cipher Designer
Date: Sun, 18 Oct 1998 04:07:00 GMT
From: Lloyd Miller <millerl@cuugnet.cuug.ab.ca>
Message-ID: <908683620.523852@server.cuug.ab.ca>
References: <jgfunj-1710982234000001@dialup175.itexas.net>
Newsgroups: sci.crypt
Lines: 25

W T Shaw <jgfunj@EnqvbSerrGrknf.pbz> wrote:
: In article <36292906.1151332@news.visi.com>, schneier@counterpane.com
: (Bruce Schneier) wrote:

:> This was in the October CRYPTO-GRAM, but I thought I'd run it through
:> sci.crypt, since so many people seem to be asking questions on the
:> topic.
:>
:> Bruce.
: ....

...
: If you are overly addicted to the opinons of certain people, you tend to
: acquire their prejudices; afterwards, know that discovering any flaws is
: prohibited, and severly punished by excommunication, which has always been
: a religious act aimed at the unfaithful so as to humiliate and silence
: them. This technique is often used as well against those that do not buy
: the bit up front.

Bruce's religion makes a lot more sense to me than your's.

--
Lloyd Miller, Calgary
millerl@cuug.ab.ca.
Terminal Insomniac

Subject: Re: Memo to the Amateur Cipher Designer
Date: Sun, 18 Oct 1998 09:02:32 -0400
From: "Jay Holovacs" <holovacs@idt.net>
Message-ID: <70cs7t$kja@nnrp1.farm.idt.net>
References: <jgfunj-1710982234000001@dialup175.itexas.net>
Newsgroups: sci.crypt
Lines: 56

W T Shaw wrote in message ...
>>>
>
>There have been many such discussions which marry some good advice with
>propaganda, serving the status quo rather than being inclusive of all
>attempts at improvement in the condition of man. A contrived obstacle
>course means being sure that few can finish, and more are discouraged from
>even trying. Those that do run the gauntlet and break the tape seem to
>confirm its validity to the blinded faithful, not withstanding the best
>intentions of those who would sit in judgement, doing the best they can to
>feel that the whole process is of inordinate value.
> [...etc...]

Newton said 'if I have seen farther than most, it is because I stood on the shoulders of giants.' It has also been said 'he who will not learn from the past is doomed to repeat it.' Bruce makes a great deal of sense. Crypto is not a random shot in the dark, it has a long history of mistakes and discoveries. Just as the patent office became littered with the products of inventors of 'perpetual energy machines' not realizing what was wrong with their great ideas, the crypto world is littered with schemes that mean nothing.

You can't get far in chemistry without learning theory and experience of those that went before. If you want to develop your own winning racing car, you'd best begin by working with as many of the machines built by other great builders as possible. Crypto is no different. If you can't break codes that are out there, why should anyone believe that you have an answer. (In truth, analysis is probably the more important part of the field now, even though most beginners want to rush in and create their own encryption algorithms.)

There is this mythology that by *not* learning how something is done, you can come up with a radical new approach. Quaint, but it doesn't work in the real world. Einstein learned existing physics before he shattered the boundaries of the known physics world. Good writers, painters and composers need to know all the rules of their art before they can break them successfully. Only in areas where there is no history of prior art can someone really come out of the blue and change things (as with small computers 15-20 years ago). Crypto is not one of those areas.

Bruce offered some really good advice for getting yourself listened to, break known codes and write up your results. These are not hard to get published. If someone who can demonstrably analyze codes produces one, there is much more reason to take such a person seriously.

Don't make excuses. Don't blame the 'establishment' that's out to stop you. Listen to people who actually know something. Prove yourself if you want to believed.

Jay

Subject: Re: Memo to the Amateur Cipher Designer
Date: Sun, 18 Oct 1998 11:34:03 -0600
From: jgfunj@EnqvbSerrGrknf.pbz (W T Shaw)
Message-ID: <jgfunj-1810981134030001@dialup122.itexas.net>
References: <70cs7t$kja@nnrp1.farm.idt.net>
Newsgroups: sci.crypt
Lines: 76

In article <70cs7t$kja@nnrp1.farm.idt.net>, "Jay Holovacs"
<holovacs@idt.net> wrote:
>
> Bruce offered some really good advice for getting yourself listened to,
> break known codes and write up your results. These are not hard to get
> published. If someone who can demonstrably analyze codes produces one, there
> is much more reason to take such a person seriously.
>
> Don't make excuses. Don't blame the 'establishment' that's out to stop you.
> Listen to people who actually know something. Prove yourself if you want to
> believed.
>
The big question is what does one actually know from knowledge delivered in a transfusion. In the days when some of us started working, there were skant few resources to work with, and no open debate on any current crypto advances. That time was distasteful, and we should not go there in any respect.

Science is less about belief and more about evidence. You seem to confuse the two. You might prejudice your results by looking for the wrong

evidence. In the end, each observation stands or falls on its own through replication and not by the clout of a sole documenter. Personalities can get involved, but true inquisitiveness should cause everyone to rise above that. Apprenticeships are not a universal requirement.

There is not real establishment in crypto anymore, just truth where you find it. In Bruce's work, there are sinful omissions and comissions, but the subject is so large that this would always be a surity in some form. To judge his character, we will see if he mentions in the future any things he has previously ignored and have been pointed out directly to him. If he is a true scientist, he will include such. I would gamble that he in the end will chose fairness. You should not figure that he is doomed fail to rise to that imperative.

We each have the option of presenting contasting and contradictory evidence as we see it. Look for the amount of cryptological information to explode as growth occurs in a myrid of directions. No one person will be able to keep it under his thumb, and we better be willing to accept increased specialization as it does.

It might surprise you that I do considerable work in code breaking, not necessarily the ones you would choose. Sometimes I am more successful, sometimes less. The goal for me is to learn how to defeat a weakness and apply it in a refined design. To broadcast prematurely such results would give others the advantage in future designs that I might reserve for myself; and so probably it is with others.

It does not follow that a successful analysis can always to a better design, and particularily that one known for solving a particular problem can pose a better one. For some it is more important to learn from failures and move on to something better than to trash anothers work as a justification for raising a consultant fee.

Back to Bruce, he has a couple of interesting designs in a relatively narrow defined area of crypto. He is also a good researcher and has assembled a certain amount of material in a convenient form. He is a serious organizer, and exercises great concentration to get what he wants. He is an excellent presenter, and most capable in matters closely related to his work. He can be a bear in his zeal, and he can be most cheerful when receiving complements, we all tend to be that way at such times. He defends his work as he should; it is considerable, showing a colossal amount of labor, be it like anything else pushing certain viewpoints over others.

He is worthy of some respect and will continue to inspire lots of people. But, because he is a limited human being, it also follows that the percentage of cryptography he understands will continue to slip as the field outpaces anyones ablility to completely grasp it. This is not a discourteous observation, just another real one. It could be as well said for all others, even those who are into their work as a priority. We should all be humbled by the magnitude of the that problem.
--

Insanity means doing the same thing over and over again and expecting different results...like CDA2.

Decrypt with ROT13 to get correct email address.

Subject: Re: Memo to the Amateur Cipher Designer
Date: Sun, 18 Oct 1998 22:32:14 GMT
From: dscott@networkusa.net
Message-ID: <70dq9ejjt1@nnrp1.dejanews.com>
References: <70cs7t$kja@nnrp1.farm.idt.net>
Newsgroups: sci.crypt
Lines: 48

In article <70cs7t$kja@nnrp1.farm.idt.net>,
  "Jay Holovacs" <holovacs@idt.net> wrote:
>
> W T Shaw wrote in message ...
> >>>
> >
> >There have been many such discussions which marry some good advice with
> >propaganda, serving the status quo rather than being inclusive of all
> >attempts at improvement in the condition of man. A contrived obstacle
> >course means being sure that few can finish, and more are discouraged from
> >even trying. Those that do run the gauntlet and break the tape seem to
> >confirm its validity to the blinded faithful, not withstanding the best
> >intentions of those who would sit in judgement, doing the best they can to
> >feel that the whole process is of inordinate value.
> > [...etc...]
>
> Newton said 'if I have seen farther than most, it is because I stood on the
> shoulders of giants.' It has also been said 'he who will not learn from the
> past is doomed to repeat it.' Bruce makes a great deal of sense. Crypto is
> not a random shot in the dark, it has a long history of mistakes and
> discoveries. Just as the patent office became littered with the products of
> inventors of 'perpetual energy machines' not realizing what was wrong with
> their great ideas, the crypto world is littered with schemes that mean
> nothing.
>
> You can't get far in chemistry without learning theory and experience of
> those that went before. If you want to develop your own winning racing car,

> you'd best begin by working with as many of the machines built by other
> great builders as possible. Crypto is no different. If you can't break codes
> that are out there, why should anyone believe that you have an answer. (In
> truth, analysis is probably the more important part of the field now, even
> though most beginners want to rush in and create their own
> encryption algorithms.)
>

I like your chemsitry example it fits well witht the load of stuff
Bruce is trying to pass off. In chemistry when I had it in school we
got to see a lovely film on the Noble gases. A bunch of PHD experts
siad lets try to make compounds useing this part of periodic table. They
do all sorts of brainy exotic things. But no compounds formed from the
Noble gases. At end of film they pompously stated how foolish it was
to even try and that there are no such compounds. Then are teacher
should us the articles how some nobodys made some. Yes the chemistry
was a good example.

----------== Posted via Deja News, The Discussion Network ==----------
http://www.dejanews.com/ Search, Read, Discuss, or Start Your Own

---

Subject: Re: Memo to the Amateur Cipher Designer
Date: 19 Oct 1998 02:14:15 GMT
From: jsavard@freenet.edmonton.ab.ca ()
Message-ID: <70e79n$896$1@news.sas.ab.ca>
References: <70cs7t$kja@nnrp1.farm.idt.net>
Newsgroups: sci.crypt
Lines: 35

Jay Holovacs (holovacs@idt.net) wrote:
: Newton said 'if I have seen farther than most, it is because I stood on the
: shoulders of giants.' It has also been said 'he who will not learn from the
: past is doomed to repeat it.' Bruce makes a great deal of sense. Crypto is
: not a random shot in the dark, it has a long history of mistakes and
: discoveries.

I certainly do agree with this, people wanting to design a new cipher
ought to be familiar with what has gone before.

: Bruce offered some really good advice for getting yourself listened to,
: break known codes and write up your results. These are not hard to get
: published. If someone who can demonstrably analyze codes produces one, there
: is much more reason to take such a person seriously.

Well, I certainly have to admit there is truth to that. In _two_ ways.

Certainly, a cipher design from someone like Eli Biham, one of the
academic discoverers of differential cryptanalysis, is going to be taken
seriously, as it should.

And a general familiarity with the principles of cryptanalysis, especially
as they apply to the kind of cipher one is attempting to design, is going
to be an important guide away from various pitfalls.

However, cryptanalysis is a discipline of its own, and requires either
considerable stamina or advanced mathematical skills. One does not quite
need these qualifications to design a secure cipher, particularly if one
is following your earlier advice and not ignoring the lessons of previous
designs.

Of course, if one wants a hearing, if one's qualifications are modest, one
should be modest.

John Savard

<jgfunj-1710982234000001@dialup175.itexas.net>
<70cs7t$kja@nnrp1.farm.idt.net> <70e79n$896$1@news.sas.ab.ca>
Cache-Post-Path: cnn!unknown@spike.long.harlequin.co.uk

Subject: Re: Memo to the Amateur Cipher Designer
Date: 19 Oct 1998 14:29:21 +0100
From: Mark Tillotson <markt@harlequin.co.uk>
Message-ID: <kxsogkfzny.fsf@harlequin.co.uk>
References: <36292906.1151332@news.visi.com>
Newsgroups: sci.crypt
Lines: 64

jsavard@freenet.edmonton.ab.ca () wrote:
| And a general familiarity with the principles of cryptanalysis, especially
| as they apply to the kind of cipher one is attempting to design, is going
| to be an important guide away from various pitfalls.
|
| However, cryptanalysis is a discipline of its own, and requires either
| considerable stamina or advanced mathematical skills. One does not quite
| need these qualifications to design a secure cipher, particularly if one
| is following your earlier advice and not ignoring the lessons of previous
| designs.

Nonsense! How on earth can you claim to design a secure cipher if you are
incapable of distinquishing a weak cipher from a strong cipher??? It
just doesn't make any sense at all.

That's like saying a blind person can paint a scene in correct colours despite being unable to see what they are doing! Sure it's not _impossible_ that it could happen, but no-one with an ounce of common sense expects such an outrageously lucky outcome (or even for the paint to end up on the canvas!!) We don't want a cipher that might well be extremely strong, we want ciphers that are extremely likely to be strong...

With cipher design we don't even have a way of distinquishing strong from weak, we merely have techniques or varying sophistication for trying to identify and measure weakness, and people more or less highly skilled at applying them and inventing new techniques of analysis. The cipher designer needs to iterate the design through more and more sophisticated analyses until it _seems_ both appropriately secure and efficient. Then the next step is to enlist some more people to help in the process of searching for missed weaknesses, and eventually publication.

Its an ongoing process of weeding out weaknesses, gradually bringing in more and more people as one's confidence in the lack of "silly mistakes" grows, just like any other safety-critical large-scale engineering project.

There certainly is a lot of scope for amateurs to suggest _ideas_ to use in cipher design, but a serious _design_ itself needs to be at the centre of such a process of cryptanalysis, not just made up by inspired guesswork.

So I'd agree that experience in cryptanalysis isn't necessary to create a plausible _looking_ design, but that it is an _absolute necessity_ for creating an actual publishable design (unless you just wanted to create a toy cipher). If the 10000's of amateur cryptographers all started publishing designs, we'd be in a total mess!

These days ciphers are expected to be used as building blocks for all sorts of security primitives, so even "security" involves resisitance to many different modes of attack, and the amount of work needed to design a cipher is usually beyond the skills and patience of a single individual anyway.

Our whole digital infrastructure is going to depend on future ciphers being secure, and I for one don't want to see the information superhighway made of "concrete" that's washes away the first time it rains because its recipe was formulated by a well-meaning amateur who didn't know anything about QA'ing concrete!!

__Mark
[markt@harlequin.co.uk | http://www.harlequin.co.uk/ | +44(0)1954 785433]

Subject: Re: Memo to the Amateur Cipher Designer
Date: Thu, 22 Oct 1998 19:13:05 GMT
From: jsavard@tenMAPSONeerf.edmonton.ab.ca (John Savard)
Message-ID: <362f81e7.14525013@news.prosurfr.com>
References: <kxsogkfzny.fsf@harlequin.co.uk>
Newsgroups: sci.crypt
Lines: 31

Mark Tillotson <markt@harlequin.co.uk> wrote, in part:
>jsavard@freenet.edmonton.ab.ca () wrote:

>| However, cryptanalysis is a discipline of its own, and requires either
>| considerable stamina or advanced mathematical skills. One does not quite
>| need these qualifications to design a secure cipher, particularly if one
>| is following your earlier advice and not ignoring the lessons of previous
>| designs.

>Nonsense! How on earth can you claim to design a secure cipher if you are
>_incapable_ of distinquishing a weak cipher from a strong cipher??? It
>just doesn't make any sense at all.

I emphatically _agree_ that if you know *nothing* about cryptanalysis, you won't be able to design a secure cipher (except by accident, or by copying someone else's design with trivial changes).

I thought, though, that I was being clear in what I was trying to say; that while a _knowledge_ of cryptanalysis is needed, actually being a cryptanalyst - actually being able to carry out, in full, the cryptanalysis of a difficult cipher, or being able to make theoretical contributions to the field - is not, strictly speaking, necessary (although Bruce is still right that those sorts of qualifications will get you taken seriously) to design a secure cipher.

Maybe you would find that position wrong-headed too, and I can understand that. But it's not nearly the same as the position you correctly characterized as expecting a blind person to paint.

John Savard
http://members.xoom.com/quadibloc/index.html

Subject: Re: Memo to the Amateur Cipher Designer
Date: Thu, 22 Oct 1998 13:56:59 -0600

From: jgfunj@EnqvbSerrGrknf.pbz (W T Shaw)
Message-ID: <jgfunj-2210981357000001@dialup159.itexas.net>
References: <kxsogkfzny.fsf@harlequin.co.uk>
Newsgroups: sci.crypt
Lines: 131

In article <kxsogkfzny.fsf@harlequin.co.uk>, Mark Tillotson
<markt@harlequin.co.uk> wrote:

> jsavard@freenet.edmonton.ab.ca () wrote:
> | And a general familiarity with the principles of cryptanalysis, especially
> | as they apply to the kind of cipher one is attempting to design, is going
> | to be an important guide away from various pitfalls.
> |
> | However, cryptanalysis is a discipline of its own, and requires either
> | considerable stamina or advanced mathematical skills. One does not quite
> | need these qualifications to design a secure cipher, particularly if one
> | is following your earlier advice and not ignoring the lessons of previous
> | designs.
>
> Nonsense! How on earth can you claim to design a secure cipher if you are
> _incapable_ of distinquishing a weak cipher from a strong cipher??? It
> just doesn't make any sense at all.

Many imply that if you simply follow their rules for cipher construction,
you need not do much of the analysis yourself. They even suggest that
someone else do it, a catch 22.
>
> That's like saying a blind person can paint a scene in correct colours
> despite being unable to see what they are doing! Sure it's not
> _impossible_ that it could happen, but no-one with an ounce of common sense
> expects such an outrageously lucky outcome (or even for the paint to
> end up on the canvas!!)

Did you see the story on TV about the guy who is blind and bicycles. He
has learned sonic location, and clicks his tongue as a generator.

Out of curiosity, I once asked a blind man to describe different colors.
The explanations he had remembered from what he had heard made sense. This
is somewhat in line with my above comments about following someone else's
crypto design strategies.

> We don't want a cipher that might well be
> extremely strong, we want ciphers that are extremely likely to be
> strong...

According to someone else's plan....
>
> With cipher design we don't even have a way of distinquishing strong
> from weak, we merely have techniques or varying sophistication for
> trying to identify and measure weakness, and people more or less
> highly skilled at applying them and inventing new techniques of
> analysis. The cipher designer needs to iterate....

As in a Feisal construction?

> the design through
> more and more sophisticated analyses until it _seems_ both
> appropriately secure and efficient.

Appropriate for whom? Not too strong, but just about right?

Efficient? Meets the requirements of someone of few thoughts worth
encrypting or that of a government who would hide the routine from the
prying eyes of the curious?

> Then the next step is to enlist
> some more people to help in the process of searching for missed
> weaknesses, and eventually publication.

Enlist? Easy for the military to say. Publication? Easy for the
established press to say.
>
> Its an ongoing process of weeding out weaknesses, gradually bringing
> in more and more people as one's confidence in the lack of "silly
> mistakes" grows, just like any other safety-critical large-scale
> engineering project.

Large scale projects can fail too...The Broken Pyramid, notable bridge
collapses(interior and exterior), numerous levee systems, multistory old
masonry buildings in earthquakes, anti-disease vaccinations pushed in
hopes that they would work in time of war, etc.

Granted, it is easy to guard against some cryptological mistakes, while
others are sort of obscure, overcoming prejudice and criticism against
concepts that are generally well know is also a hurdle.
>
> There certainly is a lot of scope for amateurs to suggest _ideas_ to
> use in cipher design, but a serious _design_ itself needs to be at the
> centre of such a process of cryptanalysis, not just made up by
> inspired guesswork.

All productive guesswork is inspired, it is just the nature of the
inspiration that you really question, but it does not always come in the
same form. If you do follow someone else's ingredient list, you may, no

surprise, produce ideas in line with the common logic of that receipe.
>
> So I'd agree that experience in cryptanalysis isn't necessary to
> create a plausible _looking_ design, but that it is an _absolute
> necessity_ for creating an actual publishable design (unless you just
> wanted to create a toy cipher). If the 10000's of amateur
> cryptographers all started publishing designs, we'd be in a total mess!

Speak for yourself white man.
>
> These days ciphers are expected to be used as building blocks for all
> sorts of security primitives, so even "security" involves resisitance
> to many different modes of attack, and the amount of work needed to
> design a cipher is usually beyond the skills and patience of a single
> individual anyway.

Ah, beyond the Expert Syndrome to the group-think phenomena. And, I
suppose that such a design system would put ALL the names of the
contributers out front. It would seem best to acknowledge even the most
meager of efforts that helped the team, as it might make a difference if
the coffee was brewed correctly. Including all the help would make the
front people look less important, or are they not the essential ingredient
in the first place?
>
> Our whole digital infrastructure is going to depend on future ciphers
> being secure, and I for one don't want to see the information
> superhighway made of "concrete" that's washes away the first time it
> rains because its recipe was formulated by a well-meaning amateur who
> didn't know anything about QA'ing concrete!!
>
Roads unlike cryptographic algorithms are best built under the old Roman
model, and pavement has not improved much since. The problem with the
whole digital infrastucture is that we have a very sick patient and the
base question should be whether we should start over beginning with the
very design of the lowest end to include historically known security
wisdom and exted it throughout, not to whether we can put it in a rest
home so as to prolong the agony.
--

Passing a budgit that no single person has fully seen is bad. Ronnie was right at
least once.

Decrypt with ROT13 to get correct email address.

Subject: Re: Memo to the Amateur Cipher Designer
Date: Mon, 26 Oct 1998 03:41:23 GMT
From: schneier@counterpane.com (Bruce Schneier)
Message-ID: <3633eed3.1151576@news.visi.com>
References: <jgfunj-2210981357000001@dialup159.itexas.net>
Newsgroups: sci.crypt
Lines: 40

On Thu, 22 Oct 1998 13:56:59 -0600, jgfunj@EnqvbSerrGrknf.pbz (W T
Shaw) wrote:
>Many imply that if you simply follow their rules for cipher construction,
>you need not do much of the analysis yourself. They even suggest that
>someone else do it, a catch 22.

Many are wrong.

>> That's like saying a blind person can paint a scene in correct colours
>> despite being unable to see what they are doing! Sure it's not
>> _impossible_ that it could happen, but no-one with an ounce of common sense
>> expects such an outrageously lucky outcome (or even for the paint to
>> end up on the canvas!!)
>
>Did you see the story on TV about the guy who is blind and bicycles. He
>has learned sonic location, and clicks his tongue as a generator.
>
>Out of curiosity, I once asked a blind man to describe different colors.
>The explanations he had remembered from what he had heard made sense. This
>is somewhat in line with my above comments about following someone else's
>crypto design strategies.

Remember that security is orthogonal to functionality. A blind guy
gets feedback--from the pavement, large objects, etc--to tell him he
is succeeding or failing at bicycle riding. An algorithm designer
gets no such feedback.

>> We don't want a cipher that might well be
>> extremely strong, we want ciphers that are extremely likely to be
>> strong...
>
>According to someone else's plan....

The totality of "someone elses" are the attackers.

Bruce
****
Bruce Schneier, President, Counterpane Systems Phone: 612-823-1098
101 E Minnehaha Parkway, Minneapolis, MN 55419 Fax: 612-823-1590
Free crypto newsletter. See: http://www.counterpane.com

Subject: Re: Memo to the Amateur Cipher Designer
Date: Sun, 25 Oct 1998 23:31:04 -0600
From: jgfunj@EnqvbSerrGrknf.pbz (W T Shaw)
Message-ID: <jgfunj-2510982331040001@207.22.198.192>
References: <3633eed3.1151576@news.visi.com>
Newsgroups: sci.crypt
Lines: 20

In article <3633eed3.1151576@news.visi.com>, schneier@counterpane.com
(Bruce Schneier) wrote:
>
> Remember that security is orthogonal to functionality. A blind guy
> gets feedback--from the pavement, large objects, etc--to tell him he
> is succeeding or failing at bicycle riding. An algorithm designer
> gets no such feedback.

Sure he does if and when what he did is discovered to be wanting.
However, it is an oft used tactic to hide that news so that you can
continue to read his mail.

More to the point, the AES process is *designed* as a big feedback
mechanism, the quicker acting the better.
>
--

Please excuse if there are multiple postings for my responses...I have no idea where
they come from as I only receive one confirmation for each posting from my
newsserver.

Decrypt with ROT13 to get correct email address.

Subject: Re: Memo to the Amateur Cipher Designer
Date: Mon, 26 Oct 1998 03:38:23 GMT
From: schneier@counterpane.com (Bruce Schneier)
Message-ID: <3633ee7c.1064691@news.visi.com>
References: <kxsogkfzny.fsf@harlequin.co.uk>
Newsgroups: sci.crypt
Lines: 77

On 19 Oct 1998 14:29:21 +0100, Mark Tillotson <markt@harlequin.co.uk>
wrote:

>jsavard@freenet.edmonton.ab.ca () wrote:
>| And a general familiarity with the principles of cryptanalysis, especially
>| as they apply to the kind of cipher one is attempting to design, is going
>| to be an important guide away from various pitfalls.
>|
>| However, cryptanalysis is a discipline of its own, and requires either
>| considerable stamina or advanced mathematical skills. One does not quite
>| need these qualifications to design a secure cipher, particularly if one
>| is following your earlier advice and not ignoring the lessons of previous
>| designs.
>
>Nonsense! How on earth can you claim to design a secure cipher if you are
>_incapable_ of distinquishing a weak cipher from a strong cipher??? It
>just doesn't make any sense at all.
>
>That's like saying a blind person can paint a scene in correct colours
>despite being unable to see what they are doing! Sure it's not
>_impossible_ that it could happen, but no-one with an ounce of common sense
>expects such an outrageously lucky outcome (or even for the paint to
>end up on the canvas!!) We don't want a cipher that might well be
>extremely strong, we want ciphers that are extremely likely to be
>strong...

Good comment.

>With cipher design we don't even have a way of distinquishing strong
>from weak, we merely have techniques or varying sophistication for
>trying to identify and measure weakness, and people more or less
>highly skilled at applying them and inventing new techniques of
>analysis. The cipher designer needs to iterate the design through
>more and more sophisticated analyses until it _seems_ both
>appropriately secure and efficient. Then the next step is to enlist
>some more people to help in the process of searching for missed
>weaknesses, and eventually publication.
>
>Its an ongoing process of weeding out weaknesses, gradually bringing
>in more and more people as one's confidence in the lack of "silly
>mistakes" grows, just like any other safety-critical large-scale
>engineering project.
>
>There certainly is a lot of scope for amateurs to suggest _ideas_ to
>use in cipher design, but a serious _design_ itself needs to be at the
>centre of such a process of cryptanalysis, not just made up by
>inspired guesswork.

Agreed.

>So I'd agree that experience in cryptanalysis isn't necessary to
>create a plausible _looking_ design, but that it is an _absolute
>necessity_ for creating an actual publishable design (unless you just

>wanted to create a toy cipher). If the 10000's of amateur
>cryptographers all started publishing designs, we'd be in a total mess!

1000s of TriStratas and Ultimate Privacies. Sounds horrible.

>These days ciphers are expected to be used as building blocks for all
>sorts of security primitives, so even "security" involves resisitance
>to many different modes of attack, and the amount of work needed to
>design a cipher is usually beyond the skills and patience of a single
>individual anyway.
>
>Our whole digital infrastructure is going to depend on future ciphers
>being secure, and I for one don't want to see the information
>superhighway made of "concrete" that's washes away the first time it
>rains because its recipe was formulated by a well-meaning amateur who
>didn't know anything about QA'ing concrete!!

Rah rah.

Bruce
****
Bruce Schneier, President, Counterpane Systems Phone: 612-823-1098
101 E Minnehaha Parkway, Minneapolis, MN 55419 Fax: 612-823-1590
Free crypto newsletter. See: http://www.counterpane.com

Subject: Re: Memo to the Amateur Cipher Designer
Date: Mon, 26 Oct 1998 08:18:40 GMT
From: cryptonews@my-dejanews.com
Message-ID: <711b90$he8$1@nnrp1.dejanews.com>
References: <3633ee7c.1064691@news.visi.com>
Newsgroups: sci.crypt
Lines: 30

In article <3633ee7c.1064691@news.visi.com>,
  schneier@counterpane.com (Bruce Schneier) wrote:
> >So I'd agree that experience in cryptanalysis isn't necessary to
> >create a plausible _looking_ design, but that it is an _absolute
> >necessity_ for creating an actual publishable design (unless you just
> >wanted to create a toy cipher). If the 10000's of amateur
> >cryptographers all started publishing designs, we'd be in a total mess!
>
> 1000s of TriStratas and Ultimate Privacies. Sounds horrible.

    This is not about crypto and security, it is rather becoming about
    Bruce Schneir BIG EGO and what he thinks the world should be.

    You should be ashemed of posting this response on SCI.CRYPT.

Cheers,

Sam Kamille

> Rah rah.
>
> Bruce
> **********
> Bruce Schneier, President, Counterpane Systems Phone: 612-823-1098
> 101 E Minnehaha Parkway, Minneapolis, MN 55419 Fax: 612-823-1590
> Free crypto newsletter. See: http://www.counterpane.com
>

----------== Posted via Deja News, The Discussion Network ==----------
http://www.dejanews.com/ Search, Read, Discuss, or Start Your Own

---

Subject: Re: Memo to the Amateur Cipher Designer
Date: Mon, 26 Oct 1998 13:06:03 GMT
From: dscott@networkusa.net
Message-ID: <711s3r$3j4$1@nnrp1.dejanews.com>
References: <711b90$he8$1@nnrp1.dejanews.com>
Newsgroups: sci.crypt
Lines: 33

In article <711b90$he8$1@nnrp1.dejanews.com>,
  cryptonews@my-dejanews.com wrote:
> In article <3633ee7c.1064691@news.visi.com>,
> schneier@counterpane.com (Bruce Schneier) wrote:
> > >So I'd agree that experience in cryptanalysis isn't necessary to
> > >create a plausible _looking_ design, but that it is an _absolute
> > >necessity_ for creating an actual publishable design (unless you just
> > >wanted to create a toy cipher). If the 10000's of amateur
> > >cryptographers all started publishing designs, we'd be in a total mess!
> >
> > 1000s of TriStratas and Ultimate Privacies. Sounds horrible.
>
> This is not about crypto and security, it is rather becoming about
> Bruce Schneir BIG EGO and what he thinks the world should be.
>
> You should be ashemed of posting this response on SCI.CRYPT.
>
> Cheers,
>

> Sam Kamille
>
>
    Play it again Sam. For a while I thought I was the only one
intelligent enough to notice Mr B.S. is nothing but a big BLOWHART
it seems that every one else was following him like a god. If you
read my hate mail messages.

--
http://cryptography.org/cgi-bin/crypto.cgi/Misc/scott19u.zip
ftp search for scott*u.zip in norway

----------== Posted via Deja News, The Discussion Network ==----------
http://www.dejanews.com/ Search, Read, Discuss, or Start Your Own

---

Subject: Re: Memo to the Amateur Cipher Designer
Date: Mon, 26 Oct 1998 18:00:16 GMT
From: jsavard@tenMAPSONeerf.edmonton.ab.ca (John Savard)
Message-ID: <3634b729.7043376@news.prosurfr.com>
References: <711b90$he8$1@nnrp1.dejanews.com>
Newsgroups: sci.crypt
Lines: 39

cryptonews@my-dejanews.com wrote, in part:
>In article <3633ee7c.1064691@news.visi.com>,
> schneier@counterpane.com (Bruce Schneier) wrote:

>> >So I'd agree that experience in cryptanalysis isn't necessary to
>> >create a plausible _looking_ design, but that it is an _absolute
>> >necessity_ for creating an actual publishable design (unless you just
>> >wanted to create a toy cipher). If the 10000's of amateur
>> >cryptographers all started publishing designs, we'd be in a total mess!

>> 1000s of TriStratas and Ultimate Privacies. Sounds horrible.

> This is not about crypto and security, it is rather becoming about
> Bruce Schneir BIG EGO and what he thinks the world should be.

> You should be ashemed of posting this response on SCI.CRYPT.

No, that is not at all true or fair.

I'll admit, I'm a bit more liberal.

I think that, while some knowledge of cryptanalysis is needed to
design a secure cipher, one doesn't actually need the level of
knowledge that one can use to easily prove you know what you're
talking about - and so life is more complicated.

I'd also say that amateur cipher designs are harmless enough, if the
person responsible is reasonably modest, and doesn't try to claim he
has the solution to everybody's problem, and all other ciphers are
irrelevant.

Actually, if there were 10,000 amateur cipher designs published, the
harm would be mainly to amateur cipher designers - in that their
designs would recieve even less attention than is now the case. The
channels of professional publication would simply become a bit more
exclusive - in self-defence, to remain usable, not out of egotism.

John Savard
http://members.xoom.com/quadibloc/index.html

Subject: Re: Memo to the Amateur Cipher Designer
Date: Wed, 28 Oct 1998 18:32:41 GMT
From: aquiranx@goliat.ugr.es (Gurripato (x=nospam))
Message-ID: <363758d1.27371552@news.cica.es>
References: <3634b729.7043376@news.prosurfr.com>
Newsgroups: sci.crypt
Lines: 22

On Mon, 26 Oct 1998 18:00:16 GMT, jsavard@tenMAPSONeerf.edmonton.ab.ca (John
Savard) wrote:

>
>Actually, if there were 10,000 amateur cipher designs published, the
>harm would be mainly to amateur cipher designers - in that their
>designs would recieve even less attention than is now the case. The
>channels of professional publication would simply become a bit more
>exclusive - in self-defence, to remain usable, not out of egotism.
>
>John Savard
>http://members.xoom.com/quadibloc/index.html

Not to speak of crypto-credibility as a whole. If those 10.000
amateur cipher existed and were published, crypto vendors would start
incorporating them into their products. How would the customers react when
9.990 of those ciphers are proved to be weak? They would distrust all
ciphers in general, and perhaps turn into some "credible" source like the
USGov or the NSA (or Bill Gates, come to that). Designing homemade ciphers

is fun; pretending they are strong and useful in real life is another
matter.

Subject: Re: Memo to the Amateur Cipher Designer
Date: Wed, 28 Oct 1998 19:13:43 GMT
From: ritter@io.com (Terry Ritter)
Message-ID: <36376ccf.5706026@news.io.com>
References: <363758d1.27371552@news.cica.es>
Newsgroups: sci.crypt
Lines: 65

On Wed, 28 Oct 1998 18:32:41 GMT, in <363758d1.27371552@news.cica.es>,
in sci.crypt aquiranx@goliat.ugr.es (Gurripato (x=nospam)) wrote:

>On Mon, 26 Oct 1998 18:00:16 GMT, jsavard@tenMAPSONeerf.edmonton.ab.ca (John
>Savard) wrote:
>
>
>>
>>Actually, if there were 10,000 amateur cipher designs published, the
>>harm would be mainly to amateur cipher designers - in that their
>>designs would recieve even less attention than is now the case. The
>>channels of professional publication would simply become a bit more
>>exclusive - in self-defence, to remain usable, not out of egotism.
>>
>>John Savard
>>http://members.xoom.com/quadibloc/index.html
>
> Not to speak of crypto-credibility as a whole. If those 10.000
>amateur cipher existed and were published, crypto vendors would start
>incorporating them into their products. How would the customers react when
>9.990 of those ciphers are proved to be weak? They would distrust all
>ciphers in general, and perhaps turn into some "credible" source like the
>USGov or the NSA (or Bill Gates, come to that). Designing homemade ciphers
>is fun; pretending they are strong and useful in real life is another
>matter.

This is a legitimate concern, but it applies to everything we have.

The problem is that we cannot measure the strength of a cipher. But
that means *any* cipher, even the well-regarded ones.

So, if one of the few well-regarded ciphers that people actually use
is found weak, does this not reflect on the entire field, the whole
profession, indeed the whole concept of cryptography?

I would argue that the better situation for "crypto-credibility" is if
we have many ciphers and various users need to select a cipher on
their own, and thus take some responsibility for it. In a "many
cipher" environment, if a particular cipher fails, some subset of the
population is affected, and they quickly change to another cipher.
But if the major cipher we all use fails, and users do not normally
change ciphers (and thus probably can't), we have a major disaster for
a long time, and *that* is how we lose "crypto-credibility." Better a
lot of small failures and short changeovers than one huge failure with
a changeover that could take years.

And when a cipher is actually *found* weak, this is actually the
lesser problem, *provided* users have alternate ciphers *and* can
select them rather quickly. The larger problem is when a cipher *is*
weak and none of our guys can show that. Then we get to use that
thing. It is *dangerous* for everybody to use the same cipher.

Analysis cannot be enough. We also need to establish defensive
protocols -- such as the ability to change ciphers, universal
multi-ciphering, and having "many ciphers" (to reduce the value of the
traffic under any one) -- to help mitigate our fundamental uncertainty
about cipher strength.

Terry Ritter ritter@io.com http://www.io.com/~ritter/
Crypto Glossary http://www.io.com/~ritter/GLOSSARY.HTM

Subject: Re: Memo to the Amateur Cipher Designer
Date: 28 Oct 1998 14:52:00 -0500
From: juola@mathcs.duq.edu (Patrick Juola)
Message-ID: <717sl0$m32$1@quine.mathcs.duq.edu>
References: <36376ccf.5706026@news.io.com>
Newsgroups: sci.crypt
Lines: 41

In article <36376ccf.5706026@news.io.com>, Terry Ritter <ritter@io.com> wrote:
>
>On Wed, 28 Oct 1998 18:32:41 GMT, in <363758d1.27371552@news.cica.es>,
>in sci.crypt aquiranx@goliat.ugr.es (Gurripato (x=nospam)) wrote:
>
>>On Mon, 26 Oct 1998 18:00:16 GMT, jsavard@tenMAPSONeerf.edmonton.ab.ca (John
>>Savard) wrote:
>>

>>
>>>
>>>Actually, if there were 10,000 amateur cipher designs published, the
>>>harm would be mainly to amateur cipher designers - in that their
>>>designs would recieve even less attention than is now the case. The
>>>channels of professional publication would simply become a bit more
>>>exclusive - in self-defence, to remain usable, not out of egotism.
>>>
>>>John Savard
>>>http://members.xoom.com/quadibloc/index.html
>>
>> Not to speak of crypto-credibility as a whole. If those 10.000
>>amateur cipher existed and were published, crypto vendors would start
>>incorporating them into their products. How would the customers react when
>>9.990 of those ciphers are proved to be weak? They would distrust all
>>ciphers in general, and perhaps turn into some "credible" source like the
>>USGov or the NSA (or Bill Gates, come to that). Designing homemade ciphers
>>is fun; pretending they are strong and useful in real life is another
>>matter.
>
>This is a legitimate concern, but it applies to everything we have.
>
>

>The problem is that we cannot measure the strength of a cipher. But
>that means *any* cipher, even the well-regarded ones.

This is untrue. It's fairly easy to come up with a measurement
of the strength of a cypher -- and even a fairly meaningful measurement
of as an upper bound of the strength of a cypher -- to wit, no cypher
can be stronger than the effort required by the best known attack
to break it.

-kitten

Subject: Re: Memo to the Amateur Cipher Designer
Date: Mon, 02 Nov 1998 17:57:07 GMT
From: ritter@io.com (Terry Ritter)
Message-ID: <363df228.1199990@news.io.com>
References: <717sl0$m32$1@quine.mathcs.duq.edu>
Newsgroups: sci.crypt
Lines: 31

On 28 Oct 1998 14:52:00 -0500, in <717sl0$m32$1@quine.mathcs.duq.edu>,
in sci.crypt juola@mathcs.duq.edu (Patrick Juola) wrote:

>>[...]
>>The problem is that we cannot measure the strength of a cipher. But
>>that means *any* cipher, even the well-regarded ones.
>
>This is untrue. It's fairly easy to come up with a measurement
>of the strength of a cypher -- and even a fairly meaningful measurement
>of as an upper bound of the strength of a cypher -- to wit, no cypher
>can be stronger than the effort required by the best known attack
>to break it.

From the user's standpoint, an upper bound is *not* the strength, and
is not even a useful estimate.

For a user, a *lower* bound would be acceptable, since an Opponent
would have to invest that amount of effort *at least* to penetrate the
cipher. But an *upper* bound is inherently deceptive of the effort an
Opponent might have to spend. The real value could be much, much
less. For any upper bound, the real strength could be none at all.

To the user, since we have *neither* the real strength, *nor* the
lower bound, we have no useful measure of strength at all.

Terry Ritter ritter@io.com http://www.io.com/~ritter/
Crypto Glossary http://www.io.com/~ritter/GLOSSARY.HTM

Subject: Re: Memo to the Amateur Cipher Designer
Date: 2 Nov 1998 15:23:18 -0500
From: juola@mathcs.duq.edu (Patrick Juola)
Message-ID: <71l4bm$rm9$1@quine.mathcs.duq.edu>
References: <363df228.1199990@news.io.com>
Newsgroups: sci.crypt
Lines: 39

In article <363df228.1199990@news.io.com>, Terry Ritter <ritter@io.com> wrote:
>
>On 28 Oct 1998 14:52:00 -0500, in <717sl0$m32$1@quine.mathcs.duq.edu>,
>in sci.crypt juola@mathcs.duq.edu (Patrick Juola) wrote:
>
>>>[...]
>>>The problem is that we cannot measure the strength of a cipher. But
>>>that means *any* cipher, even the well-regarded ones.
>>
>>This is untrue. It's fairly easy to come up with a measurement
>>of the strength of a cypher -- and even a fairly meaningful measurement

>>of as an upper bound of the strength of a cypher -- to wit, no cypher
>>can be stronger than the effort required by the best known attack
>>to break it.
>
>From the user's standpoint, an upper bound is *not* the strength, and
>is not even a useful estimate.

Depends on which user you talk to, I suspect. It's certainly a useful estimate if the upper bound is too small to represent an acceptable risk. In other words, people *know* not to use DES not because of the outside chance that a brilliant cryptographer might be able to crack it quickly, but because there's no possible way that it could resist a determined brute-force attempt.

One can, after all, always buy insurance against the lucky break.

>To the user, since we have *neither* the real strength, *nor* the
>lower bound, we have no useful measure of strength at all.

Again, this is incorrect. I stand by my original statement that we have a meaningful measure. Just because it doesn't do what *YOU* want doesn't make it nonexistent. Mere dislike has rarely been able to conjure things out of existence.

I have an upper bound, I insure against the lower bound being smaller than I envision, and the risk becomes Lloyd's.

-kitten

Subject: Re: Memo to the Amateur Cipher Designer
Date: Tue, 10 Nov 1998 04:37:50 GMT
From: ritter@io.com (Terry Ritter)
Message-ID: <3647c318.8268430@news.io.com>
References: <71l4bm$rm9$1@quine.mathcs.duq.edu>
Newsgroups: sci.crypt
Lines: 51

On 2 Nov 1998 15:23:18 -0500, in <71l4bm$rm9$1@quine.mathcs.duq.edu>,
in sci.crypt juola@mathcs.duq.edu (Patrick Juola) wrote:

>[...]
>In other words, people *know* not to use DES
>not because of the outside chance that a brilliant cryptographer
>might be able to crack it quickly, but because there's no possible
>way that it could resist a determined brute-force attempt.

When cryptanalysis identifies a practical break, it provides very useful information.

But most cryptanalysis does not do this, but instead produces yet another impractical break. The user thus gets to judge between ciphers with impractical breaks and ciphers as yet unanalyzed. Cryptanalysis does not provide information useful for making such a decision.

>>To the user, since we have *neither* the real strength, *nor* the
>>lower bound, we have no useful measure of strength at all.
>
>Again, this is incorrect. I stand by my original statement that
>we have a meaningful measure. Just because it doesn't do what
>*YOU* want doesn't make it nonexistent. Mere dislike has rarely
>been able to conjure things out of existence.

Not only does cryptanalysis not do what *I* want, it hardly does anything at all *unless* it comes up with a practical break. The vast majority of cryptanalysis -- so praised by so many -- does nothing at all to inform users about the strength of their cipher.

Indeed, The Opponents may be superior to our analysts in many ways, and may have breaks our guys do not. What our guys find in no way implies that The Opponents have nothing better: that is the crux of the problem.

>I have an upper bound, I insure against the lower bound being
>smaller than I envision, and the risk becomes Lloyd's.

So if you have an affair, and The Opponents provide your wife with that information, does Lloyds guarantee a new wife, one just as good or better?

Terry Ritter ritter@io.com http://www.io.com/~ritter/
Crypto Glossary http://www.io.com/~ritter/GLOSSARY.HTM

Subject: Re: Memo to the Amateur Cipher Designer
Date: Tue, 10 Nov 1998 16:49:40 GMT
From: jsavard@tenMAPSONeerf.edmonton.ab.ca (John Savard)
Message-ID: <36486db8.1953802@news.prosurfr.com>

References: <3647c318.8268430@news.io.com>
Newsgroups: sci.crypt
Lines: 22

ritter@io.com (Terry Ritter) wrote, in part:

>When cryptanalysis identifies a practical break, it provides very
>useful information.

>But most cryptanalysis does not do this, but instead produces yet
>another impractical break. The user thus gets to judge between
>ciphers with impractical breaks and ciphers as yet unanalyzed.
>Cryptanalysis does not provide information useful for making such a
>decision.

Ah. Sorry for failing to understand what you were getting at: since
differential, meet-in-the-middle attacks, etc., require enormous
quantities of known plaintext, either it is not clear they invalidate
a system for practical use, or, if they do prompt some precautionary
measures, the result is still not known to be secure.

And your point that not all risks can be handled by insurance is true
and amusing.

John Savard
http://www.freenet.edmonton.ab.ca/~jsavard/index.html

Subject: Re: Memo to the Amateur Cipher Designer
Date: Tue, 10 Nov 1998 18:12:23 GMT
From: schneier@counterpane.com (Bruce Schneier)
Message-ID: <364880f0.803066@news.visi.com>
References: <3647c318.8268430@news.io.com>
Newsgroups: sci.crypt
Lines: 34

On Tue, 10 Nov 1998 04:37:50 GMT, ritter@io.com (Terry Ritter) wrote:
>When cryptanalysis identifies a practical break, it provides very
>useful information.
>
>But most cryptanalysis does not do this, but instead produces yet
>another impractical break. The user thus gets to judge between
>ciphers with impractical breaks and ciphers as yet unanalyzed.
>Cryptanalysis does not provide information useful for making such a
>decision.

To many of us, impractical breaks provide very useful information to
judge between ciphers.

>Not only does cryptanalysis not do what *I* want, it hardly does
>anything at all *unless* it comes up with a practical break. The vast
>majority of cryptanalysis -- so praised by so many -- does nothing at
>all to inform users about the strength of their cipher.

Probably. But to me, that's because users are not mathematicians.
The vast majority of cryptoanalysis does a lot of inform
cryptographers about the strength of ciphers.

There's an NSA saying: "Attacks always get better." Ciphers that
allow theoretical breaks are weaker than ciphers that don't. For
example, there is an attack against IDEA the works against 4.5 round
variants. If there were a cipher for which that other attack did not
work, then ALL OTHER THINGS BEING EQUAL I would prefer that other
cipher to IDEA.

Bruce
****
Bruce Schneier, President, Counterpane Systems   Phone: 612-823-1098
101 E Minnehaha Parkway, Minneapolis, MN 55419   Fax: 612-823-1590
           Free crypto newsletter. See: http://www.counterpane.com

---

Subject: Re: Memo to the Amateur Cipher Designer
Date: Wed, 11 Nov 1998 15:18:01 GMT
From: "Douglas A. Gwyn" <DAGwyn@null.net>
Message-ID: <3649AA9A.E8719FD7@null.net>
References: <364880f0.803066@news.visi.com>
Newsgroups: sci.crypt
Lines: 27

Bruce Schneier wrote:
> To many of us, impractical breaks provide very useful information to
> judge between ciphers.

They provide information, which you may *choose* to use in judging,
but that is not necessarily a rational choice. To be rational, its
relevance to the functional criteria needs to be established.

> There's an NSA saying: "Attacks always get better." Ciphers that
> allow theoretical breaks are weaker than ciphers that don't.

Ah, but how do you know that they don't? Unless you have a proof
of that, instead what you have is a lack of knowledge of any

successful method of attack. That doesn't mean one cannot exist.

> For
> example, there is an attack against IDEA the works against 4.5 round
> variants. If there were a cipher for which that other attack did not
> work, then ALL OTHER THINGS BEING EQUAL I would prefer that other
> cipher to IDEA.

For that to be rational, you'd need to demonstrate that all other
things are indeed equal. But that is most unlikely!

I think, as often happens in academia, attention is focused too
heavily on areas where metrics exist, whether or not the metrics
have practical value.

Subject: Re: Memo to the Amateur Cipher Designer
Date: Thu, 12 Nov 1998 21:59:21 GMT
From: ritter@io.com (Terry Ritter)
Message-ID: <364b5a27.14538834@news.io.com>
References: <364880f0.803066@news.visi.com>
Newsgroups: sci.crypt
Lines: 71

On Tue, 10 Nov 1998 18:12:23 GMT, in <364880f0.803066@news.visi.com>,
in sci.crypt schneier@counterpane.com (Bruce Schneier) wrote:

>On Tue, 10 Nov 1998 04:37:50 GMT, ritter@io.com (Terry Ritter) wrote:
>[...]
>>Not only does cryptanalysis not do what *I* want, it hardly does
>>anything at all *unless* it comes up with a practical break. The vast
>>majority of cryptanalysis -- so praised by so many -- does nothing at
>>all to inform users about the strength of their cipher.
>
>Probably. But to me, that's because users are not mathematicians.
>The vast majority of cryptoanalysis does a lot of inform
>cryptographers about the strength of ciphers.

"Strength" is usually taken to be the minimum possible effort used for
any possible successful attack.

Finding a successful attack certainly tells us that "strength" can be
no higher than that attack. But it does not tell us what the strength
really is. So the attack tells us *nothing* about the real strength
of the cipher.

I would think it quite odd indeed that any mathematician would say
otherwise.

>There's an NSA saying: "Attacks always get better."

We might just as well say: "Any cipher a man can make, another can
break." Which means *any* cipher is vulnerable.

These sayings have their place: Nobody is going to break a cipher by
starting out saying that the job cannot be done. Nobody is going to
improve an attack by starting out thinking the first attack is the
final word. Such sayings have their place in encouraging creative
cryptanalysis on apparently very tough ciphers.

But sayings are not a basis for scientific comparison.

>Ciphers that
>allow theoretical breaks are weaker than ciphers that don't.

And that is precisely the leap I have been discussing. I am aware of
no scientific basis for that statement. This takes us back to
witchcraft and old-wives-tales.

>For
>example, there is an attack against IDEA the works against 4.5 round
>variants. If there were a cipher for which that other attack did not
>work, then ALL OTHER THINGS BEING EQUAL I would prefer that other
>cipher to IDEA.

And that is a different argument. That is the extrapolation argument
with which I agree.

The argument with which I disagree is that a cipher *with* an
impractical break (and which cannot reasonably be extrapolated to
further weakness) can be considered weaker than a cipher *without* an
impractical break.

To the extent that cryptanalysis produces impractical breaks, that
work tells us nothing about the practical strength of ciphers.

Terry Ritter ritter@io.com http://www.io.com/~ritter/
Crypto Glossary http://www.io.com/~ritter/GLOSSARY.HTM

Subject: Re: Memo to the Amateur Cipher Designer
Date: Tue, 03 Nov 1998 00:28:01 GMT
From: sandy.harris@sympatico.ca (Sandy Harris)
Message-ID: <l6s%1.110$GK.251745@news20.bellglobal.com>
References: <363df228.1199990@news.io.com>
Newsgroups: sci.crypt
Lines: 60

ritter@io.com (Terry Ritter) wrote:

>in sci.crypt juola@mathcs.duq.edu (Patrick Juola) wrote:
>
>>>[...]
>>>The problem is that we cannot measure the strength of a cipher. But
>>>that means *any* cipher, even the well-regarded ones.
>>
>>This is untrue. It's fairly easy to come up with a measurement
>>of the strength of a cypher -- and even a fairly meaningful measurement
>>of as an upper bound of the strength of a cypher -- to wit, no cypher
>>can be stronger than the effort required by the best known attack
>>to break it.
>
>From the user's standpoint, an upper bound is *not* the strength, and
>is not even a useful estimate.
>
>For a user, a *lower* bound would be acceptable, since an Opponent
>would have to invest that amount of effort *at least* to penetrate the
>cipher. But an *upper* bound is inherently deceptive of the effort an
>Opponent might have to spend. The real value could be much, much
>less. For any upper bound, the real strength could be none at all.
>
>To the user, since we have *neither* the real strength, *nor* the
>lower bound, we have no useful measure of strength at all.

Basically, I think you're right here. But I have a question.

We can in fact take the minimum of a set of upper bounds derived
from all the obvious attacks.

```
        Brute force search.
        Meet-in-the middle search if that appears possible.
        Linear & differential cryptanalysis.
        An attempt to write the cipher as a system of
                Boolean equations expressing ciphertext bits in
                terms of key & plaintext and then, given a bunch
                of plaintext/ciphertext pairs, solve for the key.
        For stream ciphers, linear complexity.
        Attacks based on cycles in block ciphers.
        . . .
```

I think that for good ciphers, lower bounds on the resources required
for most or all of those can be proved. Any lower bound on resources
needed for an attack is also an upper bound on the strength of the
cipher. It cannot be stronger overall than it is against that attack.

If all of those are much higher than our worst-case estimate of
attacker's resources, then we still don't know the strength of
the cipher, but we do at least know that:

```
        unless the cipher has a weakness not tested above, it
                is strong enough
        if it does have such a weakness, an attacker is going to
                have to be clever, lucky and/or persistent to find it
        only a new attack based on an unknown weakness can
                succeed
```

This still does not measure the real strength, but it at least
gives us some reason to hope.

Subject: Re: Memo to the Amateur Cipher Designer
Date: Tue, 03 Nov 1998 09:01:24 GMT
From: dscott@networkusa.net
Message-ID: <71mgp3$f3h$1@nnrp1.dejanews.com>
References: <l6s%1.110$GK.251745@news20.bellglobal.com>
Newsgroups: sci.crypt
Lines: 58

In article <l6s%1.110$GK.251745@news20.bellglobal.com>,
  sandy.harris@sympatico.ca (Sandy Harris) wrote:
> ritter@io.com (Terry Ritter) wrote:
>
> >in sci.crypt juola@mathcs.duq.edu (Patrick Juola) wrote:
> >
> >>>[...]
> >>>The problem is that we cannot measure the strength of a cipher. But
> >>>that means *any* cipher, even the well-regarded ones.
> >>
> >>This is untrue. It's fairly easy to come up with a measurement
> >>of the strength of a cypher -- and even a fairly meaningful measurement
> >>of as an upper bound of the strength of a cypher -- to wit, no cypher
> >>can be stronger than the effort required by the best known attack

```
> >>to break it.
> >
> >From the user's standpoint, an upper bound is *not* the strength, and
> >is not even a useful estimate.
> >
> >For a user, a *lower* bound would be acceptable, since an Opponent
> >would have to invest that amount of effort *at least* to penetrate the
> >cipher.  But an *upper* bound is inherently deceptive of the effort an
> >Opponent might have to spend.  The real value could be much, much
> >less.  For any upper bound, the real strength could be none at all.
> >
> >To the user, since we have *neither* the real strength, *nor* the
> >lower bound, we have no useful measure of strength at all.
>
> Basically, I think you're right here. But I have a question.
>
> We can in fact take the minimum of a set of upper bounds derived
> from all the obvious attacks.
>
>         Brute force search.
>         Meet-in-the middle search if that appears possible.
>         Linear & differential cryptanalysis.
>         An attempt to write the cipher as a system of
>                 Boolean equations expressing ciphertext bits in
>                 terms of key & plaintext and then, given a bunch
>                 of plaintext/ciphertext pairs, solve for the key.
>         For stream ciphers, linear complexity.
>         Attacks based on cycles in block ciphers.
>         . . .
       Some other things that most miss that should be added
to this is how much information is needed by the guy breaking
to know if he his decoded the file. This may same like a hard
to follow concept but if one needs only to like at a small fragment
of file to runs tests to check for a solution then it is a measureble
weakness. My method in scottNu was designed to eliminate this
weakness that is in all the Fishy des type of ciphers.


--
http://cryptography.org/cgi-bin/crypto.cgi/Misc/scott19u.zip
ftp search for scott*u.zip in norway

------------== Posted via Deja News, The Discussion Network ==----------
http://www.dejanews.com/       Search, Read, Discuss, or Start Your Own
```

```
Subject: Re: Memo to the Amateur Cipher Designer
Date: 3 Nov 1998 16:48:57 GMT
From: jmccarty@sun1307.spd.dsccc.com (Mike McCarty)
Message-ID: &LT;71nc5p$6br$1@relay1.dsccc.com>
References: &LT;l6s%1.110$GK.251745@news20.bellglobal.com>
Newsgroups: sci.crypt
Lines: 59

In article &LT;l6s%1.110$GK.251745@news20.bellglobal.com>,
Sandy Harris &LT;sandy.harris@sympatico.ca> wrote:
)Basically, I think you're right here. But I have a question.
)
)We can in fact take the minimum of a set of upper bounds derived
)from all the obvious attacks.
)
)        Brute force search.
)        Meet-in-the middle search if that appears possible.
)        Linear & differential cryptanalysis.
)        An attempt to write the cipher as a system of
)                Boolean equations expressing ciphertext bits in
)                terms of key & plaintext and then, given a bunch
)                of plaintext/ciphertext pairs, solve for the key.
)        For stream ciphers, linear complexity.
)        Attacks based on cycles in block ciphers.
)        . . .

Are you including attacks based on, say, bribery? Unless you are
willing specifically to state the exact list of attacks (which seems
undesireable), then you must state a specific criterion by which one
may, by application of the criterion, determine whether any given
proposed attack falls in the list of canonical attacks. This seems
difficult to me.

)I think that for good ciphers, lower bounds on the resources required
)for most or all of those can be proved. Any lower bound on resources
)needed for an attack is also an upper bound on the strength of the
)cipher. It cannot be stronger overall than it is against that attack.

This principle seems good to me.

)If all of those are much higher than our worst-case estimate of
)attacker's resources, then we still don't know the strength of
)the cipher, but we do at least know that:
)
)        unless the cipher has a weakness not tested above, it
)                is strong enough
)        if it does have such a weakness, an attacker is going to
)                have to be clever, lucky and/or persistent to find it
```

```
)        only a new attack based on an unknown weakness can
)                succeed
)
)This still does not measure the real strength, but it at least
)gives us some reason to hope.
```

If we can devise some predicate P(.) which can be applied to attacks and which determines whether the proposed attack satisfies the predicate for canonicity, then I think your idea is workable. It seems to me that formulating this predicate will be (unless it is in the form of a list of canonical attacks) very difficult to do. Perhaps not impossible. This looks to me to be a reasonable research area.

Mike
--
---
char *p="char *p=%c%s%c;main(){printf(p,34,p,34);}";main(){printf(p,34,p,34);}
This message made from 100% recycled bits.
I don't speak for Alcatel <- They make me say that.

Subject: Re: Memo to the Amateur Cipher Designer
Date: Tue, 03 Nov 1998 17:05:48 GMT
From: ritter@io.com (Terry Ritter)
Message-ID: <363f37b9.3868604@news.io.com>
References: <l6s%1.110$GK.251745@news20.bellglobal.com>
Newsgroups: sci.crypt
Lines: 86

On Tue, 03 Nov 1998 00:28:01 GMT, in
<l6s%1.110$GK.251745@news20.bellglobal.com>, in sci.crypt
sandy.harris@sympatico.ca (Sandy Harris) wrote:

```
>ritter@io.com (Terry Ritter) wrote:
>>[...]
>>To the user, since we have *neither* the real strength, *nor* the
>>lower bound, we have no useful measure of strength at all.
>
>Basically, I think you're right here. But I have a question.
>
>We can in fact take the minimum of a set of upper bounds derived
>from all the obvious attacks.
>
>        Brute force search.
>        Meet-in-the middle search if that appears possible.
>        Linear & differential cryptanalysis.
>        An attempt to write the cipher as a system of
>                Boolean equations expressing ciphertext bits in
>                terms of key & plaintext and then, given a bunch
>                of plaintext/ciphertext pairs, solve for the key.
>        For stream ciphers, linear complexity.
>        Attacks based on cycles in block ciphers.
>        . . .
>
>I think that for good ciphers, lower bounds on the resources required
>for most or all of those can be proved.
```

Yet if we look in the cryptanalytic literature, we almost invariably find a *sequence* of ever-better versions that each improve on the previous attack. I believe I have seen improved versions of:

* Meet-in-the-Middle,
* Linear Cryptanalysis, and
* Differential Cryptanalysis,

but I think such sequences are common.

Now, if it were practical to know lower bounds for these attacks, why would we ever see improved versions in the literature? And since we *do* see improved versions, how can we believe in computing lower bounds for strength, even for a particular attack?

```
>Any lower bound on resources
>needed for an attack is also an upper bound on the strength of the
>cipher. It cannot be stronger overall than it is against that attack.
>
>If all of those are much higher than our worst-case estimate of
>attacker's resources, then we still don't know the strength of
>the cipher, but we do at least know that:
>
>        unless the cipher has a weakness not tested above, it
>                is strong enough
```

These "attacks" each depend upon human interpretation. Now who tests the tester? If someone tells us that a cipher is strong under these attacks, how can we believe it?

```
>        if it does have such a weakness, an attacker is going to
>                have to be clever, lucky and/or persistent to find it
```

We use cryptography to face attackers with far greater resources in

training, experience, equipment, time and motivation. Just because
we have failed to find a weakness is no reason to think the
attackers will also. There is no correlation, no correct
extrapolation.

> only a new attack based on an unknown weakness can
> succeed
>
>This still does not measure the real strength, but it at least
>gives us some reason to hope.

The reason for hope is the acknowledgement of the problem and the use
of protocols which tend to minimize it. The strength quality is
literally out of control, so we cannot trust that.

Terry Ritter ritter@io.com http://www.io.com/~ritter/
Crypto Glossary http://www.io.com/~ritter/GLOSSARY.HTM

Subject: Re: Memo to the Amateur Cipher Designer
Date: 6 Nov 1998 08:42:08 GMT
From: olson@umbc.edu (Bryan G. Olson; CMSC (G))
Message-ID: <71ucp0ghp1@news.umbc.edu>
References: <363f37b9.3868604@news.io.com>
Newsgroups: sci.crypt
Lines: 23

Terry Ritter (ritter@io.com) wrote:

[...]
: We use cryptography to face attackers with far greater resources in
: training, experience, equipment, time and motivation. Just because
: *we* have failed to find a weakness is no reason to think the
: attackers will also. There is no correlation, no correct
: extrapolation.

No correlation???? You've talked yourself into a bunch of
nonsense.

Note that for no correlation to exist, it is necessary
that no cipher is weakness free. If any is, then both
defender and attacker must fail to find weakness and
therefor there would be a correlation. So how do you
know there's no correlation?

--Bryan

Subject: Re: Memo to the Amateur Cipher Designer
Date: Mon, 02 Nov 1998 20:01:18 GMT
From: jsavard@tenMAPSONeerf.edmonton.ab.ca (John Savard)
Message-ID: <363e0e04.12569961@news.prosurfr.com>
References: <717sl0$m32$1@quine.mathcs.duq.edu>
Newsgroups: sci.crypt
Lines: 20

juola@mathcs.duq.edu (Patrick Juola) wrote, in part:

>This is untrue. It's fairly easy to come up with a measurement
>of the strength of a cypher -- and even a fairly meaningful measurement
>of as an upper bound of the strength of a cypher -- to wit, no cypher
>can be stronger than the effort required by the best known attack
>to break it.

But Terry Ritter is right that there's no easy way to derive the
actual strength (or, for that matter, a _lower_ bound on the strength
of a cipher, IMO). He feels this is an important problem in
cryptography to which not enough attention is being devoted.

I feel, on the other hand, that this isn't a problem one *can* work on
specifically. That this is a goal which requires every great question
in mathematics to have been answered. So, in a way, *all* mathematical
work proceeds to that goal - but it's a very distant one.

John Savard
http://members.xoom.com/quadibloc/index.html

Subject: Re: Memo to the Amateur Cipher Designer
Date: Tue, 03 Nov 1998 17:04:48 GMT
From: ritter@io.com (Terry Ritter)
Message-ID: <363f37a3.3846779@news.io.com>
References: <363e0e04.12569961@news.prosurfr.com>
Newsgroups: sci.crypt
Lines: 49

On Mon, 02 Nov 1998 20:01:18 GMT, in
<363e0e04.12569961@news.prosurfr.com>, in sci.crypt
jsavard@tenMAPSONeerf.edmonton.ab.ca (John Savard) wrote:

>juola@mathcs.duq.edu (Patrick Juola) wrote, in part:
>
>>This is untrue. It's fairly easy to come up with a measurement
>>of the strength of a cypher -- and even a fairly meaningful measurement
>>of as an upper bound of the strength of a cypher -- to wit, no cypher
>>can be stronger than the effort required by the best known attack
>>to break it.
>
>But Terry Ritter is right that there's no easy way to derive the
>actual strength (or, for that matter, a _lower_ bound on the strength
>of a cipher, IMO). He feels this is an important problem in
>cryptography to which not enough attention is being devoted.

Having no lower bounds for strength may be "an important problem" to
the academic study of cryptography.

But it also calls into question *the entire field* of practical
cryptography.

The whole point of the actual use of cryptography is to *enforce*
security. Without at least a minimum value for strength, the user has
no guarantee -- or even a useful probability -- of that.

We can try to improve this situation by multi-ciphering and other
protocols, but we have a real problem that should be universally
recognized and commonly discussed. This is not just, or even mainly,
"academic," it is a real problem in practice for real systems.

>I feel, on the other hand, that this isn't a problem one *can* work on
>specifically. That this is a goal which requires every great question
>in mathematics to have been answered. So, in a way, *all* mathematical
>work proceeds to that goal - but it's a very distant one.

The actual truth of not knowing the strength of the ciphers we field
for people to use is not just an academic problem. Breaking a cipher
gives us no more useful information about the strength of the cipher
than we had before that cipher was broken.

Terry Ritter ritter@io.com http://www.io.com/~ritter/
Crypto Glossary http://www.io.com/~ritter/GLOSSARY.HTM

Subject: Re: Memo to the Amateur Cipher Designer
Date: Tue, 03 Nov 1998 23:42:10 GMT
From: jsavard@tenMAPSONeerf.edmonton.ab.ca (John Savard)
Message-ID: <363f8efc.24854596@news.prosurfr.com>
References: <363f37a3.3846779@news.io.com>
Newsgroups: sci.crypt
Lines: 74

ritter@io.com (Terry Ritter) wrote, in part:
>jsavard@tenMAPSONeerf.edmonton.ab.ca (John Savard) wrote:

>>I feel, on the other hand, that this isn't a problem one *can* work on
>>specifically. That this is a goal which requires every great question
>>in mathematics to have been answered. So, in a way, *all* mathematical
>>work proceeds to that goal - but it's a very distant one.

>The actual truth of not knowing the strength of the ciphers we field
>for people to use is not just an academic problem. Breaking a cipher
>gives us no more useful information about the strength of the cipher
>than we had before that cipher was broken.

Breaking a cipher can, of course, give us information that is useful
in a negative sense: we can know for a fact that a certain cipher is
too weak to be worth using.

Of course it isn't merely an academic problem, but something desirable
from a practical point of view, to have a _lower_ bound for a cipher's
strength. Solving it perfectly, in the academic sense, however,
appears to be impossible - unless mathematics ever gets "finished".

Since the current rough-and-ready method of taking the upper bound
with a grain of salt and discounting it as a strength guess is not
acceptable, I assume you want a real, provable, lower bound.

And whether one uses it for academic or practical purposes, it's just
as unobtainable. I don't contradict your statement that this is a
serious problem for cryptography that we don't have this: but if there
is no realistic prospect of obtaining it, directed effort at finding a
way of obtaining lower bounds on cipher strength, however badly we
need it, is _still_ a waste of time.

The fact that we all grow old, and this inevitably leads to death, is
certainly a serious problem; but until very recently, attempting to
solve this problem was still not a rational act.

Of course, the last time I said this, shortly after I came up with an "insight" into cryptanalysis that I thought got us *slightly* closer to the goal; no proven lower bound, but at least a little bit more insight for our guesses.

I'll put that insight on the record again: on a very high level, cryptanalysis can be divided into three types of operation:

- Brute force trying of all possibilities for the key or for some part of the key;

- Directly calculating the key from other information (e.g. calculating the private key from the public key by factoring; trying a probable word on a Vigenere);

- Separating the key - or, and this is very important, some internal transform of the key - into pieces that can be brute-forced separately.

I claim that #3 is sufficiently broad and vague to cover 99% of all cryptanalytic techniques in existence - yet it has enough content to suggest ways of making ciphers stronger, and maybe even is a first step to quantifying strength - in an imperfect and incomplete sense.

Perhaps what I'm saying is obvious to you, and the reason you are going beyond stating the fact that lower bounds on cryptographic strength don't exist to criticizing the cryptographic community for their nonexistence is because you do have some insight into how one might begin to go about looking for a way to find lower bounds.

If you do have such an insight, you have come up with something of great value.

John Savard
http://members.xoom.com/quadibloc/index.html

Subject: Re: Memo to the Amateur Cipher Designer
Date: Wed, 04 Nov 1998 11:56:18 GMT
From: "Douglas A. Gwyn" <DAGwyn@null.net>
Message-ID: <364040DF.98B1A714@null.net>
References: <363f8efc.24854596@news.prosurfr.com>
Newsgroups: sci.crypt
Lines: 16

John Savard wrote:
> I'll put that insight on the record again: on a very high level,
> cryptanalysis can be divided into three types of operation:
> - Brute force trying of all possibilities for the key or for some part
> of the key;
> - Directly calculating the key from other information (e.g.
> calculating the private key from the public key by factoring; trying a
> probable word on a Vigenere);
> - Separating the key - or, and this is very important, some internal
> transform of the key - into pieces that can be brute-forced
> separately.

It's nice to try to bring order to the subject, but the above
is not complete. Some cryptanalysis doesn't even recover the
key (this happened to me with Zendian DDHAA as I recall),
and at other times one recovers a decimation of the true key.

Subject: Re: Memo to the Amateur Cipher Designer
Date: Wed, 04 Nov 1998 23:49:49 GMT
From: dscott@networkusa.net
Message-ID: <71qp6u$a6h$1@nnrp1.dejanews.com>
References: <364040DF.98B1A714@null.net>
Newsgroups: sci.crypt
Lines: 33

In article <364040DF.98B1A714@null.net>,
  "Douglas A. Gwyn" <DAGwyn@null.net> wrote:
> John Savard wrote:
> > I'll put that insight on the record again: on a very high level,
> > cryptanalysis can be divided into three types of operation:
> > - Brute force trying of all possibilities for the key or for some part
> > of the key;
> > - Directly calculating the key from other information (e.g.
> > calculating the private key from the public key by factoring; trying a
> > probable word on a Vigenere);
> > - Separating the key - or, and this is very important, some internal
> > transform of the key - into pieces that can be brute-forced
> > separately.
>
> It's nice to try to bring order to the subject, but the above
> is not complete. Some cryptanalysis doesn't even recover the
> key (this happened to me with Zendian DDHAA as I recall),
> and at other times one recovers a decimation of the true key.
>

And sometimes the encryption program itself does not use or solve for the key that the method is based on. Like in scott19u.zip Which an anonymouse crypto person as offered to set up a site talking about it. I will fix up his omissions and misunderstandings as time goes on.

--
http://cryptography.org/cgi-bin/crypto.cgi/Misc/scott19u.zip
http://members.xoom.com/ecil/index.htm

----------== Posted via Deja News, The Discussion Network ==----------
http://www.dejanews.com/ Search, Read, Discuss, or Start Your Own

---

Subject: Re: Memo to the Amateur Cipher Designer
Date: Tue, 10 Nov 1998 03:33:24 GMT
From: ritter@io.com (Terry Ritter)
Message-ID: <3647b3ca.4349316@news.io.com>
References: <363f8efc.24854596@news.prosurfr.com>
Newsgroups: sci.crypt
Lines: 59

On Tue, 03 Nov 1998 23:42:10 GMT, in
<363f8efc.24854596@news.prosurfr.com>, in sci.crypt
jsavard@tenMAPSONeerf.edmonton.ab.ca (John Savard) wrote:

>[...]
>Breaking a cipher can, of course, give us information that is useful
>in a negative sense: we can know for a fact that a certain cipher is
>too weak to be worth using.

But many of the results in cryptanalysis do not present us with a cipher known to be unusably weak. Many of these results need impractical efforts. Does an impractical break argue for using another cipher which has no break -- when that cipher *also* has no lower bound on strength? How is that a better situation?

>[...]
>The fact that we all grow old, and this inevitably leads to death, is
>certainly a serious problem; but until very recently, attempting to
>solve this problem was still not a rational act.

We know that life exists; all we need do is prolong it. Presumably, there are many specific problems. For each one we fix, we can scientifically verify improved results.

But we do not know that cryptographic strength exists, and we cannot verify it. No matter how many problems we fix, we have no idea whether strength has improved or not. This is a distinctly different situation.

>[...]
>Perhaps what I'm saying is obvious to you, and the reason you are
>going beyond stating the fact that lower bounds on cryptographic
>strength don't exist to criticizing the cryptographic community for
>their nonexistence is because you do have some insight into how one
>might begin to go about looking for a way to find lower bounds.

Well, it is *not* obvious to me that there *cannot* be a cipher with some amount of proven strength. I am aware of no proof that all ciphers must be weak.

But the reason I brought this stuff up and stayed with it was in direct response to the recent stuff on attacking ciphers. It was suggested that cryptanalysis is the way users know the strength of their ciphers. That suggestion is false.

In reality, cryptanalysis only benefits *users* when their particular cipher is actually shown to be weak in practice *and* the user can switch to something else. Any cryptanalytic results which show impractical breaks are irrelevant to the user and essentially contribute no information about strength.

Terry Ritter ritter@io.com http://www.io.com/~ritter/
Crypto Glossary http://www.io.com/~ritter/GLOSSARY.HTM

Subject: Re: Memo to the Amateur Cipher Designer
Date: Tue, 10 Nov 1998 17:26:20 GMT
From: jsavard@tenMAPSONeerf.edmonton.ab.ca (John Savard)
Message-ID: <3648706c.2645679@news.prosurfr.com>
References: <3647b3ca.4349316@news.io.com>
Newsgroups: sci.crypt
Lines: 116

ritter@io.com (Terry Ritter) wrote, in part:
>jsavard@tenMAPSONeerf.edmonton.ab.ca (John Savard) wrote:

>>[...]
>>Breaking a cipher can, of course, give us information that is useful
>>in a negative sense: we can know for a fact that a certain cipher is
>>too weak to be worth using.

>But many of the results in cryptanalysis do not present us with a
>cipher known to be unusably weak. Many of these results need
>impractical efforts. Does an impractical break argue for using
>another cipher which has no break -- when that cipher *also* has no
>lower bound on strength? How is that a better situation?

At that point, my comment was just a minor nitpick for the sake of
strict correctness.

An impractical break is not an argument for using a completely
different cipher, but between two closely similar ciphers, the one
protected against the impractical break but not otherwise different is
likely to be stronger.

>>[...]
>>The fact that we all grow old, and this inevitably leads to death, is
>>certainly a serious problem; but until very recently, attempting to
>>solve this problem was still not a rational act.

>We know that life exists; all we need do is prolong it. Presumably,
>there are many specific problems. For each one we fix, we can
>scientifically verify improved results.

>But we do not know that cryptographic strength exists, and we cannot
>verify it. No matter how many problems we fix, we have no idea
>whether strength has improved or not. This is a distinctly different
>situation.

Looked at that way, yes. But the analogy is aimed at a different
aspect of the situation. A fog surrounds cryptographic strength, but
it is not clear that we can lift it, or where we would begin to try to
do so.

I'm not qualified to carry it out, but I wouldn't be surprised if a
competent mathematician couldn't supply a proof that "proving a cipher
strong" is equivalent to solving the halting problem. (Which may have
occurred to Alan Turing...)

>>[...]
>>Perhaps what I'm saying is obvious to you, and the reason you are
>>going beyond stating the fact that lower bounds on cryptographic
>>strength don't exist to criticizing the cryptographic community for
>>their nonexistence is because you do have some insight into how one
>>might begin to go about looking for a way to find lower bounds.

>Well, it is *not* obvious to me that there *cannot* be a cipher with
>some amount of proven strength. I am aware of no proof that all
>ciphers must be weak.

No, there certainly won't be such a proof either! And there is one
cipher with proven strength: the one-time pad, as someone is sure to
note.

But proving something about the _work factor_ required to break a
cipher requires your proof to say something about every possible
attack - based on any mathematical principle that may not even be
discovered yet. Which is the basis for my comment about the halting
problem.

Unless something can be done about this situation, while it is valid
to note its existence as a caveat, it does not invalidate the efforts
of those who are wirking within the realm of what is practical to
achieve. Yes, cryptography is still, in this area, more of an art than
an exact science. But there appear to be fundamental reasons why this
is so.

>But the reason I brought this stuff up and stayed with it was in
>direct response to the recent stuff on attacking ciphers. It was
>suggested that cryptanalysis is the way users know the strength of
>their ciphers. That suggestion is false.

It is the way they can know the little that can be known; the people
saying this are suggesting cryptanalysis in preference to nothing,
citing examples of people designing ciphers with no knowledge of
cryptanalysis, thereby making mistakes that we already know how to
avoid, and coming up with designs that are easily broken.

My response - which I still stand by, despite almost joining the "in
club" with a spurious result against Panama - is that a cipher
designer ought to have an understanding of cryptanalysis, yes, but
having an acquaintance with it and being a fully-qualified
cryptanalyst are two different things, of which only the lesser is
needed for designing ciphers. Not that a higher degree of
qualifications isn't desirable.

But doctors and nurses and pharmacists aren't expected to always be
all three; composers should have some ability to play an instrument,
and performers should understand musical theory, but one can be
first-rate at one while only indifferent at the other.

>In reality, cryptanalysis only benefits *users* when their particular
>cipher is actually shown to be weak in practice *and* the user can
>switch to something else. Any cryptanalytic results which show
>impractical breaks are irrelevant to the user and essentially
>contribute no information about strength.

Or when the cipher they might have used was shown to be weak before they used it. Some of the impractical breaks - not all of them - do hint at the possibility of a weakness that could be exploited in practice, and that, too, is of some use.

When life gives Bruce a lemon, he makes lemonade. But I'm not aware that he was pretending it was orange juice, even if the fact that it is only lemonade should perhaps be underscored a bit more than it has been.

John Savard
http://www.freenet.edmonton.ab.ca/~jsavard/index.html

Subject: Re: Memo to the Amateur Cipher Designer
Date: Wed, 04 Nov 1998 09:26:28 -0500
From: Jerry Leichter <leichter@smarts.com>
Message-ID: <36406414.3178@smarts.com>
References: <363f37a3.3846779@news.io.com>
Newsgroups: sci.crypt
Lines: 76

| Having no lower bounds for strength may be "an important problem" to
| the academic study of cryptography.
|
| But it also calls into question *the entire field* of practical
| cryptography.
|
| The whole point of the actual use of cryptography is to *enforce*
| security. Without at least a minimum value for strength, the user has
| no guarantee -- or even a useful probability -- of that....

There is no proof of security for key locks, combination locks, or any other means of providing physical security. There is no proof of the security of any practical end-user software, operating system, or even hardware implementation. There is no proof that any source of random numbers is really random.

Even if you *had* a provably-strong encryption algorithm as an abstract mathematical object (and, in fact, we do - a true OTP), it would be impossible for you to realize it in a real world without relying on components about which you could prove very little if anything.

Almost nothing in the real world is amenable to proof in any mathematical sense. At best, we have "relative" proofs: *If* quantum mechanics is correct, *then* thermal noise from a diode is random and genuinely unpredictable. *If* our theories about how circuits work are correct, then a system built of amplifiers, samplers, and such will retain the randomness inherent in the diode's noise. *If* the real physical parts really do behave "closely enough" to our theories, then the real random noise generator really does generate random bits. And so on.

I don't want to criticize mathematical techniques. They are important in many areas, cryptography among them, because our intuitions about security aren't very good: Long experience has shown us that what seems secure on the surface may fall to very simple attacks - simple attacks that may be based on sophisticated mathematical reasoning.

But it's important to understand that ultimately *all* our knowledge of how physical artifacts work is empirical. We believe that pin-tumbler locks are reasonably secure because experience has shown that few people known how to pick them. We believe mushroom-head pins give you even more secure locks because the best lock-pickers have trouble with them. Similarly, we believe Medeco's are even more secure because no one has been able to pick them consistently.

On the other hand, Ace locks are a great example of how real attacks work: They are virtually unpickable with standard tools, but it's possible to build a special tool (I believe there's even a patent on such a tool) that makes it very easy to pick one. Since hardly anyone has one of these tools, in the real world, Ace locks are considered quite secure.

It would be really nice if there were a provably-strong cipher. It would be a triumph for mathematics. Lower bounds on complexity are known for almost no non-trivial algorithms. P vs. NP is only one part of the problem; in most interesting cases, we don't know the degree of the polynomial. We can't, in many cases, even say if a sub-exponential algorithm exists. (It's a common mistake to think that "not-P" means "exponential". There are infinitely many functions that grow faster than any polynomial but slower than any exponential. Factoring is an example of an algorithm for which no polynomial algorithm is known - but for which sub-exponential algorithms have been around for years.) Progress in this area has been slow and difficult.

When it comes to proofs (a) the history of mathematics isn't encourag-

ing: Usually, proofs are available only for approaches that are
idealized in some way to make them amenable to mathematical techniques.
These are often not particularly well suited for real-world application;
(b) even if you had such a thing, the guarantee that it could give you
concerning the entire real-world system in which it was embedded would
be so weak as to be almost useless. As always, the security of a system
is only as strong as its weakest component. With any of the
well-studied cipher systems out there today, it's unlikely that the
mathematical structure of the cipher will be the weakest component of a
real-world system in which it is embedded.

-- Jerry

Subject: Re: Memo to the Amateur Cipher Designer
Date: Fri, 06 Nov 1998 07:14:41 GMT
From: schneier@counterpane.com (Bruce Schneier)
Message-ID: <3642a1ca.765177@news.visi.com>
References: <36406414.3178@smarts.com>
Newsgroups: sci.crypt
Lines: 16

On Wed, 04 Nov 1998 09:26:28 -0500, Jerry Leichter
<leichter@smarts.com> wrote:

>As always, the security of a system
>is only as strong as its weakest component. With any of the
>well-studied cipher systems out there today, it's unlikely that the
>mathematical structure of the cipher will be the weakest component of a
>real-world system in which it is embedded.

Profoundly true.

Bruce
****
Bruce Schneier, President, Counterpane Systems Phone: 612-823-1098
101 E Minnehaha Parkway, Minneapolis, MN 55419 Fax: 612-823-1590
Free crypto newsletter. See: http://www.counterpane.com

Subject: Re: Memo to the Amateur Cipher Designer
Date: Tue, 10 Nov 1998 03:33:37 GMT
From: ritter@io.com (Terry Ritter)
Message-ID: <3647b406.4409649@news.io.com>
References: <36406414.3178@smarts.com>
Newsgroups: sci.crypt
Lines: 109

On Wed, 04 Nov 1998 09:26:28 -0500, in <36406414.3178@smarts.com>, in
sci.crypt Jerry Leichter <leichter@smarts.com> wrote:

>| Having no lower bounds for strength may be "an important problem" to
>| the academic study of cryptography.
>|
>| But it also calls into question *the entire field* of practical
>| cryptography.
>|
>| The whole point of the actual use of cryptography is to *enforce*
>| security. Without at least a minimum value for strength, the user has
>| no guarantee -- or even a useful probability -- of that....
>
>There is no proof of security for key locks, combination locks, or any
>other means of providing physical security.

But we can be more sure about these simple devices than the vast
majority of far-more-complex ciphers. I would say that locks are more
like hashes than ciphers. The only "ciphertext" in a lock is "open"
vs "close." Often, they are "keyed" by the manufacturer and there is
no key-change ability in the field. A known-plaintext attack always
works, and we accept that, while we abhor the same thing in a cipher.

But most of all, when a lock is physically broken, we will know. We
will know that someone had that capability, and exercised it.
Presumably we can use that information to improve our security. But
when a cipher is broken for real, we will *not* know. This is a much
worse and more dangerous situation.

In the physical world, we can monitor the current disposition of our
holdings and provide real-time support for attacks. We cannot do this
in the data world, so we depend more one the quality of the lock
itself. Too bad we cannot measure that quality.

>There is no proof of the
>security of any practical end-user software, operating system,

Yes. Vast complexity makes thorough testing impossible, although
proper partitioning into testable components can be a significant
improvement.

>or even
>hardware implementation.

Certainly chip manufacturers do in fact try to test every transistor and every wire in the device. This testing thus shows a very good correspondence to the schematic. Now, whether the schematic builds a device that does what we want is essentially the previous answer.

If someone gets to the chip level and can burn contacts or fuse transistors they can change the operation. But such a device will not pass its tests, so at least we have an indication of problem.

>There is no proof that any source of random
>numbers is really random.

Indeed.

>Even if you *had* a provably-strong encryption algorithm as an abstract
>mathematical object (and, in fact, we do - a true OTP), it would be
>impossible for you to realize it in a real world without relying on
>components about which you could prove very little if anything.

Then I would not *want* a proof for that type of object!

>[...]
>With any of the
>well-studied cipher systems out there today, it's unlikely that the
>mathematical structure of the cipher will be the weakest component of a
>real-world system in which it is embedded.

Well, you *say* it is "unlikely" as though you know the actual probability distribution involved. But I suspect nobody knows that, so calling it "unlikely" is really quite a leap. If we don't know, we can't make a valid statement, because we really *don't know*.

We *don't know* the strength to the cipher, so we cannot infer that it has even the strength of the surrounding system. If the cipher we use happens to be trivially weak -- provided we were twice as smart as we are -- then simply using that cipher may be the weakest link.

What I have been addressing here is I think different from relatively simple mechanical things -- most of which should be within our understanding -- and ciphers -- which seem to admit "special" understandings which produce "breaks."

The problem is these "special understandings." As long as we produce ciphers that admit new cryptanalysis, we cannot be sure of their true strength. If we cannot somehow confine or bound the unknown "special understandings," we will never have factual grounds to state that cipher strength is "unlikely" to be the weakest part of the system.

It is in this sense that cryptanalysis hurts cryptography: not because it breaks ciphers, but because we cannot know when it has done all it can do. Since we do not know that, any cipher continues to be potentially vulnerable.

Terry Ritter ritter@io.com http://www.io.com/~ritter/
Crypto Glossary http://www.io.com/~ritter/GLOSSARY.HTM

Subject: Re: Memo to the Amateur Cipher Designer
Date: Tue, 10 Nov 1998 09:29:39 -0500
From: Jerry Leichter <leichter@smarts.com>
Message-ID: <36484DD3.2452@smarts.com>
References: <3647b406.4409649@news.io.com>
Newsgroups: sci.crypt
Lines: 98

| >With any of the
| >well-studied cipher systems out there today, it's unlikely that the
| >mathematical structure of the cipher will be the weakest component of
| >a real-world system in which it is embedded.
|
| Well, you *say* it is "unlikely" as though you know the actual
| probability distribution involved. But I suspect nobody knows that,
| so calling it "unlikely" is really quite a leap. If we don't know, we
| can't make a valid statement, because we really *don't know*.

While we don't know the exact probability distribution, we *do* have empirical evidence - just as we have empirical evidence concerning the security of other systems we rely on. All we need do is look at the successful attacks of the last 20 years. Repeatedly, we find attacks against things like random number generation (key selection in Net-scape); protocols (many examples); outright incompetent use of primitives (Microsoft's use of RC4 as a stream cipher to protect password databases); physical implementations (many attacks against smart cards); and so on. Many of these attacks have had the practical effect of completely destroying the security of fielded systems.

Over the same period of time, there are *no* published attacks with any real-world significance - except for brute force relying on limited key

spaces - against any of, say, DES, RC4, IDEA, or RSA.

You can explain this difference in three ways:

1. It's just coincidence. Possible; perhaps a plausible explanation 15 years ago. But by now there are enough people, and enough attacks, that it seems very unlikely.

2. Bias in the attack distributions: If there are many more attacks against elements *other than* the mathematical structure of the algorithms than there are attacks against that structure, then naturally there will be more such successes.

   However, it's hard to believe that this is the explanation. In the same time period, we've seen many successful attacks against the mathematical structure of ad hoc cryptosystems (pkzip), secretly developed systems (some of the cell phone stuff), as well as tons and tons of attacks against systems designed by people who've had successes elsewhere and against systems that are variants of the ones that have stood the test of time. There is no evidence that those who've mounted these successful attacks have shied away from attacking the systems that remain standing. To the contrary, some of the most potent techniques (differential, linear, and related-key cryptanalysis) were developed precisely to attack DES (with little success) and PES (with enough success to lead it to be replaced by IDEA).

3. The only remaining possibility is the one I suggested: That the weakest link in current systems is most likely *not* in the mathematical structure of their encryption algorithms.

| ...What I have been addressing here is I think different from
| relatively simple mechanical things -- most of which should be within
| our understanding -- and ciphers -- which seem to admit "special"
| understandings which produce "breaks."

Do you really think that RC4, say, is any more complicated than a good combination lock? It's easy to stand here today and talk about the simplicity and transparent security of today's locks - but in fact they evolved over many years as new attacks were found, and new defenses developed. The contribution to security of many aspects of the design of a modern lock are only "obvious" when they're explained!

You can always raise the specter of the unknown techniques developed and hidden by the "black" organizations. Well, suppose I tell you that the CIA has an electromechanical device that can open any pin-tumbler lock, including the "advanced" versions like the Medeco's, within a few minutes, usually leaving no marks behind. (It uses a bunch of flexible, motor-driven probes and a combination of X-ray backscatter and ultrasound monitoring to determine pin position and alignment.) Do you still believe your analysis of the "simple, obvious" security of the locks on your doors?

Now, in fact, I made all that up. I have absolutely no idea what the CIA is capable of doing with locks. But can you provide any rational basis for claiming my "electromechanical lock picker" is any more or less likely than the NSA's secret DES crack?

I don't want to carry this analogy too far. It's a general truth that digital systems can be much more complex, and much harder to reason about, than analogue systems. This causes all kinds of risks - in security, reliability, predictability, and so on. Nevertheless, it's important to keep in mind that engineering - whether of digital or of analogue systems - is ultimately a real-world, empirical activity. Parts of it are amenable to some degree of mathematical analysis; others, at our present state of knowledge, are not; some probably will never be. Even of those amenable to analysis, the role of actual *proof* is even more limited. Where we have it, it can be very useful. But we can't wait for proofs that may, often will, never come!

-- Jerry

Subject: Re: Memo to the Amateur Cipher Designer
Date: Wed, 11 Nov 1998 15:07:27 GMT
From: "Douglas A. Gwyn" <DAGwyn@null.net>
Message-ID: <3649A820.49A28EE8@null.net>
References: <36484DD3.2452@smarts.com>
Newsgroups: sci.crypt
Lines: 19

Jerry Leichter wrote:
> Over the same period of time, there are *no* published attacks with any
> real-world significance - except for brute force relying on limited key
> spaces - against any of, say, DES, RC4, IDEA, or RSA.
> You can explain this difference in three ways:

More than three. For example:
4) Poorly designed protocols are easy to break.
5) Successful attacks against IDEA et al are likely only by

people who know *how*, and would be kept secret.

> Do you really think that RC4, say, is any more complicated than a good
> combination lock? It's easy to stand here today and talk about the
> simplicity and transparent security of today's locks - but in fact they
> evolved over many years as new attacks were found, and new defenses
> developed. The contribution to security of many aspects of the design
> of a modern lock are only "obvious" when they're explained!

That's pretty funny, to a locksmith/safeman.

Subject: Re: Memo to the Amateur Cipher Designer
Date: Wed, 11 Nov 1998 15:47:14 -0500
From: Jerry Leichter <leichter@smarts.com>
Message-ID: <3649F7D2.37AC@smarts.com>
References: <3649A820.49A28EE8@null.net>
Newsgroups: sci.crypt
Lines: 32

| > Over the same period of time, there are *no* published attacks with
| > any real-world significance - except for brute force relying on
| > limited key spaces - against any of, say, DES, RC4, IDEA, or RSA.
| > You can explain this difference in three ways:
|
| More than three. For example:

Well, OK.

| 4) Poorly designed protocols are easy to break.

This would be fully consistent with the statement that the crypto
algorithm itself is *not* the likely weak spot in the system.

| 5) Successful attacks against IDEA et al are likely only by
| people who know *how*, and would be kept secret.

I suppose. But it would take a hell of a conspiracy, given the number
of attacks against *other* systems that *have* been published.

| > Do you really think that RC4, say, is any more complicated than a
| > good combination lock? It's easy to stand here today and talk about
| > the simplicity and transparent security of today's locks - but in
| > fact they evolved over many years as new attacks were found, and new
| > defenses developed. The contribution to security of many aspects of
| > the design of a modern lock are only "obvious" when they're
| > explained!

| That's pretty funny, to a locksmith/safeman.

In what way?

-- Jerry

Subject: Re: Memo to the Amateur Cipher Designer
Date: Sun, 18 Oct 1998 22:16:30 GMT
From: dscott@networkusa.net
Message-ID: <70dpbvgal1@nnrp1.dejanews.com>
References: <36292906.1151332@news.visi.com>
Newsgroups: sci.crypt
Lines: 50

In article <36292906.1151332@news.visi.com>,
  schneier@counterpane.com (Bruce Schneier) wrote:
> This was in the October CRYPTO-GRAM, but I thought I'd run it through
> sci.crypt, since so many people seem to be asking questions on the
> topic.
>
> Bruce
>
> Memo to the Amateur Cipher Designer
>
> Congratulations. You've just invented this great new cipher, and you
> want to do something with it. You're new in the field; no one's heard
> of you, and you don't have any credentials as a cryptanalyst. You
> want to get well-known cryptographers to look at your work. What can
> you do?
>
> Unfortunately, you have a tough road ahead of you. I see about two
> new cipher designs from amateur cryptographers every week. The odds
> of any of these ciphers being secure are slim. The odds of any of
> them being both secure and efficient are negligible. The odds of any
> of them being worth actual money are virtually non-existent.
>
>

  The real truth of the matter is this. If your cipher is any good
people like Bruce will go out of there way to spread lies about it.
It is mostly a closed group of hreatless people who like it act pompous
and wave creditials about. They really know very lttle about real
crypto only the spooks at places like the NSA in america know something
about it. Part of the NSA job is to keep the world in the dark about
real ctypto. Think about it. What better way to do it than by
creating crypto preists for people to whorship. You can not get to

be a very famous person for long in real crypto with out the blessings of the NSA one way or another. Of course this is just my opionion I am running a real contest that goes to nov 11 1999 and have supplied more info they you will get in a contest from Bruce who has a lot more money at his command than I do. The most liekly reason he can't have a contest like mine is this the AES code he is trying to push is not that good. But then that is my humble opionion.

Go ahead and fell free to publish your stuff maybe Bruce will bad mouth your stuff while at the same time claiming he is to busy to have time to look at it. But he may find time to bad mouth it. Which I guess means he is afraid it is better than his stuff. One thing for sure write enough in this group and you will get spam mailed to you about his book.

----------== Posted via Deja News, The Discussion Network ==----------
http://www.dejanews.com/ Search, Read, Discuss, or Start Your Own

---

Subject: Re: Memo to the Amateur Cipher Designer
Date: Mon, 19 Oct 1998 00:47:09 -0400
From: Tim Bass <bass@silkroad.com>
Message-ID: <362AC44D.6A988225@silkroad.com>
References: <70dpbvgal1@nnrp1.dejanews.com>
Newsgroups: sci.crypt
Lines: 50

dscott@networkusa.net strangely wrote:

> The real truth of the matter is this. If your cipher is any good
> people like ***** will go out of there way to spread lies about it.
> It is mostly a closed group of hreatless people ....

Disagree. If someone does their homework and puts in the very significant time to understand iterative ciphers, understands and practices cryptanalysis, including the major art forms, then the community of those who have done their homework and have put in the significant time will not be *overly* unkind (unless of course one was having a bad day that day.)

Every person who is now "one whom has gained some fame" in this field (or any field) was once one who knew nothing about the art and science. Writing a paper which is peer reviewed takes work, hard work. Writing a book takes more work and discipline. Writing a GOOD BOOK which is well accepted by peers takes even *more* work.

Einstein was very accurate when he quipped that genius is 1 percent inspiration and 99 percent sweat.

Most of those whom have written strong ciphers did not write them without very significant research into the field. Shannon and Feistel are good places to start. Then there is a large body of literature in books and notes. I suggest all the major work in QA 76.9.A25 and then some of the Z104 areas of the stacks.

It always amazes me how the more I read, research, and study, the less and less I know!!

The peer review process is the most exciting part of professional collaboration. On the other hand, everyone appreciates those whom have done the necessary background work. It makes collaboration much more fun!!

Best Regards,

Tim
--
Tim Bass
Principal Consultant, Systems Engineering
Bass & Associates
Tel: (703) 222-4243
Fax: (703) 222-7320
EMail: bass@silkroad.com.antispam (remove antispam tag)
http://www.silkroad.com/consulting/technical.html

Subject: Re: Memo to the Amateur Cipher Designer
Date: Mon, 19 Oct 1998 11:14:49 GMT
From: dscott@networkusa.net
Message-ID: <70f6v9jud1@nnrp1.dejanews.com>
References: <362ACB2C.AEEA9007@null.net>
<362AC44D.6A988225@silkroad.com>
Newsgroups: sci.crypt
Lines: 14

In article <362ACB2C.AEEA9007@null.net>,
"Douglas A. Gwyn" <DAGwyn@null.net> wrote:
> Tim Bass wrote:
> > Einstein was very accurate when he quipped that genius is
> > 1 percent inspiration and 99 percent sweat.

>
> I think that was Edison and perspiration.
>

John you shouldn't try to confuse a Bruce Worshiper
with facts. It might confuse them.

----------== Posted via Deja News, The Discussion Network ==----------
http://www.dejanews.com/ Search, Read, Discuss, or Start Your Own

---

Subject: Re: Memo to the Amateur Cipher Designer
Date: Mon, 19 Oct 1998 12:04:29 GMT
From: david@davidham.demon.co.uk (David Hamilton)
Message-ID: <362b2aca.7134387@news.demon.co.uk>
References: <70dpbvgal1@nnrp1.dejanews.com>
Newsgroups: sci.crypt
Lines: 90

-----BEGIN PGP SIGNED MESSAGE-----

dscott@networkusa.net wrote:

>In article <36292906.1151332@news.visi.com>,
> schneier@counterpane.com (Bruce Schneier) wrote:
>> This was in the October CRYPTO-GRAM, but I thought I'd run it through
>> sci.crypt, since so many people seem to be asking questions on the
>> topic.

(snip extract)

> The real truth of the matter is this. If your cipher is any good
>people like Bruce will go out of there way to spread lies about it.

Any evidence for Bruce spreading lies? (I don't think so.)
Any evidence for your cipher being 'any good'? (nb I said 'evidence', and your opinion or assertions aren't evidence.) And don't forget, the onus is on you to provide evidence; the onus isn't on others to check your offering.

Although it is possible that somebody with very little knowledge of cryptography relevant subjects may develop a good cipher, it is unlikely that this will happen. In your case, I don't trust your cryptographic software because:

1) In the context of a dictionary attack, on 14th June, you said that you had seen a dictionary attack work on a system where the attacker never guessed the correct passphrase but he just stumbled on one that hashed to the same value. You subsequently declined to give any information about the passphrase, the hashing algorithm, the dictionary size or the method of word selection. You also declined to give the odds of stumbling on a passphrase that hashed to the same value. Your reason for declining to give this information was that the person you were referring to 'still works for the federal government'.

2) You designed all the algorithms and code used in your software. With one exception, you can't remember the names of people who 'commented'. I would suggest that 'commenting' isn't good enough anyway; what is needed is formal inspection by competent people.

(snip some)

>only the spooks at places like the NSA in america know something
>about it.

So the Chinese, Europeans and Indians are excluded. Presumably you're not a spook at a place like the USA NSA and so you don't 'know something about cryptography'. So why are you pushing your crypto software?

>Part of the NSA job is to keep the world in the dark about
>real ctypto.

Has the USA NSA succeeded in keeping you in the dark about 'real crypto'?

(snip some)

>while at the same time claiming he
(Bruce)
>is to busy to have time to look at it.

Nobody is under any obligation to look at/comment on/inspect your software. You seem to think that somebody owes you something. You've published your software, anybody who wants to use it or look at it can.

>One thing for sure write enough in this group
>and you will get spam mailed to you about his book.

I'm pretty certain I haven't been spam mailed about Bruce's book. I have seen recommendations for it and criticism of it in sci.crypt. On the other hand, I've seen a lot more ads in sci.crypt for your software.

David Hamilton. Only I give the right to read what I write and PGP allows me to make that choice. Use PGP now.
I have revoked 2048 bit RSA key ID 0x40F703B9. Please do not use. Do use:-

2048bit rsa ID=0xFA412179 Fp=08DE A9CB D8D8 B282 FA14 58F6 69CE D32D
4096bit dh ID=0xA07AEA5E Fp=28BA 9E4C CA47 09C3 7B8A CE14 36F3 3560 A07A EA5E
Both keys dated 1998/04/08 with sole UserID=<david@davidham.demon.co.uk>
-----BEGIN PGP SIGNATURE-----
Version: PGPfreeware 5.5.3i for non-commercial use <http://www.pgpi.com>
Comment: Signed with RSA 2048 bit key

iQEVAwUBNisih8o1RmX6QSF5AQHonwf6AtOdhoxumP16yzPlx7jEe2DYFInBlpMV
YR4o9wQegZlIxqw1letT2jPJijSLwih+IBLr5zViodTASmwHUXUzsOM5+wqCzZXz
1lmMxYe3JpQYDnDth+xMr6azhW/jNP+Inu4mw5vlgRzNWhcGPPhLV3kumMdApHDE
T8RfE45P8iLW58zEwwDLAXOThm7auPY4qHwC58eirZ1x26UuJZeNHzDQNm7c5bXH
HUDtIZI4s6Omw7KnXO8OXhaejBt9mrLZZZrUv1Xit7+XfimztiDUdXHf5VPJ4E98
Be3dCpA3Mdq14fqEvdvyH0nvhD2/D5KXYk7kAqAoKoCFkjMTdIIewA==
=O1EJ
-----END PGP SIGNATURE-----

Subject: Re: Memo to the Amateur Cipher Designer
Date: Mon, 19 Oct 1998 23:12:53 GMT
From: dscott@networkusa.net
Message-ID: <70gh1lgfo1@nnrp1.dejanews.com>
References: <362b2aca.7134387@news.demon.co.uk>
Newsgroups: sci.crypt
Lines: 105

In article <362b2aca.7134387@news.demon.co.uk>,
  david@davidham.demon.co.uk (David Hamilton) wrote:
> -----BEGIN PGP SIGNED MESSAGE-----
>
> dscott@networkusa.net wrote:
>
> >In article <36292906.1151332@news.visi.com>,
> > schneier@counterpane.com (Bruce Schneier) wrote:
> >> This was in the October CRYPTO-GRAM, but I thought I'd run it through
> >> sci.crypt, since so many people seem to be asking questions on the
> >> topic.
>
> (snip extract)
>
> >   The real truth of the matter is this. If your cipher is any good
> >people like Bruce will go out of there way to spread lies about it.
>
> Any evidence for Bruce spreading lies? (I don't think so.)

   Obviously you don't read all of crapola that Bruce puts out
there or you would notice some of his lies and comments in this
group about my code.
And yes I have been spamed at least twice by his for profit
company. I wrote each time for them to stop the SPAM but like
all spamers you don't even get a response.
If he hasn't spammed you feel fortunate.

> Any evidence for your cipher being 'any good'? (nb I said 'evidence', and
> your opinion or assertions aren't evidence.) And don't forget, the onus is on
> you to provide evidence; the onus isn't on others to check your offering.
>
> Although it is possible that somebody with very little knowledge of
> cryptography relevant subjects may develop a good cipher, it is unlikely that
> this will happen. In your case, I don't trust your cryptographic software
> because:
>
> 1) In the context of a dictionary attack, on 14th June, you said that you
> had seen a dictionary attack work on a system where the attacker never
> guessed the correct passphrase but he just stumbled on one that hashed to the
> same value. You subsequently declined to give any information about the
> passphrase, the hashing algorithm, the dictionary size or the method of word
> selection. You also declined to give the odds of stumbling on a passphrase
> that hashed to the same value. Your reason for declining to give this
> information was that the person you were referring to 'still works for the
> federal government'.
>
> 2) You designed all the algorithms and code used in your software. With one
> exception, you can't remember the names of people who 'commented'. I would
> suggest that 'commenting' isn't good enough anyway; what is needed is formal
> inspection by competent people.
>
> (snip some)
>
> >only the spooks at places like the NSA in america know something
> >about it.
>
> So the Chinese, Europeans and Indians are excluded. Presumably you're not a
> spook at a place like the USA NSA and so you don't 'know something about
> cryptography'. So why are you pushing your crypto software?
>
> >Part of the NSA job is to keep the world in the dark about
> >real ctypto.
>
> Has the USA NSA succeeded in keeping you in the dark about 'real crypto'?
>
> (snip some)
>
> >while at the same time claiming he

> (Bruce)
> >is to busy to have time to look at it.
>
> Nobody is under any obligation to look at/comment on/inspect your software.
> You seem to think that somebody owes you something. You've published your
> software, anybody who wants to use it or look at it can.
>
> >One thing for sure write enough in this group
> >and you will get spam mailed to you about his book.
>
> I'm pretty certain I haven't been spam mailed about Bruce's book. I have seen
> recommendations for it and criticism of it in sci.crypt. On the other hand,
> I've seen a lot more ads in sci.crypt for your software.
>
> David Hamilton. Only I give the right to read what I write and PGP allows me
> to make that choice. Use PGP now.
> I have revoked 2048 bit RSA key ID 0x40F703B9. Please do not use. Do use:-
> 2048bit rsa ID=0xFA412179 Fp=08DE A9CB D8D8 B282 FA14 58F6 69CE D32D
> 4096bit dh ID=0xA07AEA5E Fp=28BA 9E4C CA47 09C3 7B8A CE14 36F3 3560 A07A EA5E
> Both keys dated 1998/04/08 with sole UserID=<david@davidham.demon.co.uk>
> -----BEGIN PGP SIGNATURE-----
> Version: PGPfreeware 5.5.3i for non-commercial use <http://www.pgpi.com>
> Comment: Signed with RSA 2048 bit key
>
> iQEVAwUBNisih8o1RmX6QSF5AQHonwf6AtOdhoxumP16yzPlx7jEe2DYFInBlpMV
> YR4o9wQegZlIxqw1letT2jPJijSLwih+IBLr5zViodTASmwHUXUzsOM5+wqCzZXz
> 1lmMxYe3JpQYDnDth+xMr6azhW/jNP+Inu4mw5vlgRzNWhcGPPhLV3kumMdApHDE
> T8RfE45P8iLW58zEwwDLAXOThm7auPY4qHwC58eirZ1x26UuJZeNHzDQNm7c5bXH
> HUDtIZI4s6Omw7KnXO8OXhaejBt9mrLZZZrUv1Xit7+XfimztiDUdXHf5VPJ4E98
> Be3dCpA3Mdq14fqEvdvyH0nvhD2/D5KXYk7kAqAoKoCFkjMTdIIewA==
> =O1EJ
> -----END PGP SIGNATURE-----
>

----------== Posted via Deja News, The Discussion Network ==----------
http://www.dejanews.com/ Search, Read, Discuss, or Start Your Own

---

Subject: Re: Memo to the Amateur Cipher Designer
Date: Mon, 26 Oct 1998 13:15:41 GMT
From: dscott@networkusa.net
Message-ID: <711sls$4a9$1@nnrp1.dejanews.com>
References: <711fcj$m5l$1@nnrp1.dejanews.com>
  <362b2aca.7134387@news.demon.co.uk>
Newsgroups: sci.crypt
Lines: 61

In article <711fcj$m5l$1@nnrp1.dejanews.com>,
  cryptonews@my-dejanews.com wrote:
> In article <362b2aca.7134387@news.demon.co.uk>,
> david@davidham.demon.co.uk (David Hamilton) wrote:
>
> > Has the USA NSA succeeded in keeping you in the dark about 'real
> > crypto'?
>
> David,
>
> Put aside the childish temper tantrums that Bruce and his
> opponents are throwing at each other.
>
> Folks at the NSA are in the business of building strong crypto
> for preserving the USA national security. This is a valid business
> that every nation on earth is entitled to do. I am certain that
> similar folks here in the UK work as hard as the folks at the
> NSA. The NSA is also in the business of develooping efficient
> algorithm to cryptanalyze the crypto of other countries. Every
> nation that respect itself must have an NSA.
>

 What you say is true. It may even be necessary. What I don't
like is the spying on Americans for political reasons that will
someday make what the Soviet Union had look like a dream of a long
lost freedom. I don't like there role in control of our future and
the dumbing down of America. I don't like the destroying of the
Bill or Rights. I have a master degree in contorl theroy the secrect
to control is measurement. If they can read everything they can
control what we see and hear and trick is into slavery. Americns
and world citizens need free open communications without fear of
big brother reading everything or else there will be a small rich
class of people running the world and the rest of mankind will
be nothing but slaves to work and live and die in fear and controled
by the few.

> I believe that the folks at the NSA are highly respectable
> professionals just like all of us.
>
> My concern here is that when an agency like the NSA which is in
> business of National Security starts Cozying with Commercial
> developers of crypto. I doubt there is even one company in North
> America that is not in bed with the NSA.
>
> If any body's company is not doing that please let us know.
>
> Cheers,

>
> Sam
>
> ----------== Posted via Deja News, The Discussion Network ==----------
> http://www.dejanews.com/ Search, Read, Discuss, or Start Your Own
>

--
http://cryptography.org/cgi-bin/crypto.cgi/Misc/scott19u.zip
ftp search for scott*u.zip in norway

----------== Posted via Deja News, The Discussion Network ==----------
http://www.dejanews.com/ Search, Read, Discuss, or Start Your Own

---

Subject: Re: Memo to the Amateur Cipher Designer
Date: Tue, 20 Oct 1998 00:19:14 GMT
From: dscott@networkusa.net
Message-ID: <70gku1$md5$1@nnrp1.dejanews.com>
References: <362B8C89.52EDE3C@AECengineering.com>
<70dpbvgal1@nnrp1.dejanews.com>
Newsgroups: sci.crypt
Lines: 86

In article <362B8C89.52EDE3C@AECengineering.com>,
Djim <Djim@AECengineering.com> wrote:
>
>.....

>
> Mr. Scott, I would like to talk about a few of your points.
>
> 1) You don't get famous in crypto without the blessings of the NSA. While I
> cannot prove a negative, (It is just possible that the invisible hand of the
> secret masters of the crypto world has touched everyone competent), I can
> state that most of the best crypto research published is from academics and
> hobbyist crypto people. There are many people who are quite respectable
> sources of info on the internet as well of whom I am aware. I strongly
> doubt that they are all in the NSA's pocket -- especially given the efforts
> of many of them to get various patented and strong algorithms into the hands
> of the public -- and what do they gain by being pocket servants of the NSA.
> prestige? no. respect? not if it ever gets out. money? the NSA would have
> to be paying a lot of people a lot of money to keep them in their pocket
> especially with the consideration that their reputation would be ruined if
> it got out that they were an NSA shill. Something else? Maybe, but I am hard
> pressed to think what that would be.
>
what can I say I disagree with most of the above but so what.

> 2) Bruce has set a contest where anything that someone says about his cypher
> -- even some thing like this code will resist the following attack with the
> following strength, or given a reduced round version of twofish the

Actually there is no gaurantee any one will win and it appears he is the
judge of what wins if anyone does. So since it is not black and white
I doubt seriously if he is capable of honest thought. If his contest
is honest he would also offer some set of data similar to mine to
break. Since many hacker types don't think the way he does and may
have trouble writting there thoughts in way his narrow mind could
comprehend. He should if he is not chicken throw a bone to the
unwashed unclean masses. He does not so as to greatly limit the
group of people to those whose thought processes are tuned to his
own. Also if he had a contest it would be embarassing to have a
rank ametur break it.

> following properties are discovered which may be of use against the full
> version, or that certain keys have interesting properties. It is even
> possible, not probable that if nothing else is published a publication of
> commentary regarding one's opinions of the algorithm with some documentation
> of the points made could win.(Its ridiculously unlikely to happen, but the
> award is for the best paper published regarding it and its weakness/strength
> vs. attack). From what he has said a reward WILL be given. Your contest
> provides much less info -- Poorly documented code, and some limited
> plain/cyphertext pairs -- and sets the bar much higher. A full BREAK of the
> code -- nothing less will be enough. Sounds to me like no reward need ever
> be given out unless someone devoted way lots of time to this project and
> frankly most of us cannot be bothered to do so. Even 20 hours of my time
> bills for more than you offer. I feel that your code is at least secure
> enough that 50 hours of my time will not break it so why bother.
> If you are so sure of your code's security offer a real reward -- or a
> smaller prize given each - say year - for the best attacks vs your cypher.
> It would pay off in a better cypher and more respect on the group. Think
> about it.
>

I don't think it is fair or honest if I offered more cash than I could
gather up. But I think my contest is fairer in that the winner if there is
one is back and white. In short in my contest the answer is right or wrong
I have no lee way to back out like he does. However in some ways my contest
is less fair. In that I know for a fact mine is to hard to solve. You may
have realized that when you said 50 hours of your time most likely will
not solve it. Something that no one has noticed is that if I did not state
how I made the key the solution is not unique. In other words there exist

many many keys over 2**1000 that map the first plain text into the given encrypted file and some of those can unmap the second encrytped file into a file that is different by exactly 4 characters and yet they are not the same as the plain text file I started with. I am not sure if there is an easy way to type a set of 4 phrases that can map to a different solution though. I have thought about offering a 100 dollar prise to the first one who gets a close solition. That is one who comes up with just a keyraw.key file that maps first file set and then unmaps (maps) the second encrypted file into a file like the first but different by 4 characters. Is this the kind of thing you mean. Or if one finds something close to what paul onions did I could offer 100 dollars but it would have to be something that good.

 YOUR THOUGHTS WELCOME SINCE AT LEAST YOU MAY HAVE LOOKED AT IT

----------== Posted via Deja News, The Discussion Network ==----------
http://www.dejanews.com/ Search, Read, Discuss, or Start Your Own

---

Subject: Re: Memo to the Amateur Cipher Designer
Date: Tue, 20 Oct 1998 00:40:21 GMT
From: ritter@io.com (Terry Ritter)
Message-ID: <362bdbc6.3212829@news.io.com>
References: <36292906.1151332@news.visi.com>
Newsgroups: sci.crypt
Lines: 190

On Sat, 17 Oct 1998 23:35:28 GMT, in <36292906.1151332@news.visi.com>,
in sci.crypt schneier@counterpane.com (Bruce Schneier) wrote:

>[...]
>Congratulations. You've just invented this great new cipher, and you
>want to do something with it. You're new in the field; no one's heard
>of you, and you don't have any credentials as a cryptanalyst. You
>want to get well-known cryptographers to look at your work. What can
>you do?

Maybe some people are like this, but I doubt I would care to know them. If the reason for inventing a new cipher is to get someone else to look at it -- no matter who may be looking -- there would seem to be some small problem with goals.

In my view, the reason for inventing a new cipher -- like any new design -- is to deliver new advantages. If so, it advances the art, quite independent of whether the professional chorus agrees.

After that, it is up to the professional cryptographers to stay abreast of advances to the art, if they wish to continue to claim expertise. It is not the responsibility of the developers to go around and inform all the "experts" through their chosen media outlet. Either they keep up, or they are not experts on what they have missed, and it's just that simple.

>[...]
>What is hard is creating an algorithm that no one else can break, even
>after years of analysis. And the only way to prove that is to subject
>the algorithm to years of analysis by the best cryptographers around.

That last sentence is what the profession says, but it is as false, misleading, and self-delusional as anything in cryptography: Even years of analysis is not proof. It is not close to proof.

Lack of proof of weakness is not proof of strength. Obviously.

Yet we still get the same old mantra -- that every professional knows is false -- which every newbie and security officer is encouraged to believe. Why is this? Who benefits from this?

>[...]
>It's hard to get a cryptographic algorithm published.

This, of course, is simply false. It is false in the assumption that "published" means accepted by some academic journal. And it is also more or less false in that most reasonable papers *can* be shopped around and placed in some academic journal eventually. There are many journals, and each must fill a gaping maw of pages continuously.

It is true, however, that a *good* article is more than a new idea: A good article offers *insight* as opposed to mere symbology and gobbledygook. If someone provide a good presentation to something exciting and new, they won't have much trouble placing it somewhere.

>Most
>conferences and workshops won't accept designs from unknowns and
>without extensive analysis. This may seem unfair:

Not accepting designs from "unknowns" is well beyond just *seeming* unfair, it *is* unfair. It is in fact unscientific. More than that, this also has economic consequences.

Any time an academic publication will not look at, accept, and publish work based on content -- independent of its source -- that publication is spending its academic reputation. Journals exist to serve a larger ideal than to simply further the academic careers of authors -- their existence depends upon providing the best material to their readers. They can't just reject good stuff and publish the chaff without real audience consequences.

To a large extent, the same thing applies to conferences and workshops as well. Science is not tidy, and advances often do not come from those who feel they deserve to have made them. Publishers and conference leaders who do not understand this are presiding over their own decline.

>[...]
>When I started writing _Applied Cryptography_, I heard the maxim that
>the only good algorithm designers were people who spent years
>analyzing existing designs. The maxim made sense, and I believed it.

Then you were fooled. Vernam, a mere engineer in 1919: The mechanistic stream cipher, and the basis for the one time pad.

>[...]
>A cryptographer friend tells the story of an amateur who kept
>bothering him with the cipher he invented. The cryptographer would
>break the cipher, the amateur would make a change to "fix" it, and the
>cryptographer would break it again. This exchange went on a few times
>until the cryptographer became fed up. When the amateur visited him
>to hear what the cryptographer thought, the cryptographer put three
>envelopes face down on the table. "In each of these envelopes is an
>attack against your cipher. Take one and read it. Don't come back
>until you've discovered the other two attacks." The amateur was never
>heard from again.

Hell, I wouldn't go back if I *did* know the answer: Your friend is a pompous ass. That this sort of thing is ever acceptable -- let alone actually promoted in a public forum -- shows the depth to which this "profession" has sunk.

This game of "I'm better than you" is a sickness that infects the entire field of cryptography. It makes every discussion a contest, every relationship a competition, and a mockery of facts and clear, correct reasoning. It works against development in the field, and has got to go. Those professionals who are actively cultivating ego cults for their own self-gratification are part of the problem.

In the anecdote, a better alternative would be for the cryptographer to be helpful, to explain the issues, lay out a course of study, and thus in a larger sense generally address why the general public has so little understanding of this profession. We don't of course, see anecdotes about that. Why? See the above paragraph.

>[...]
>1. Describe your cipher using standard notation. This doesn't mean C
>code. There is established terminology in the literature. Learn it
>and use it; no one will learn your specialized terminology.

Yes. There are established notations for the design of logic systems, and they include both "schematics" and "flow charts" as well as C. But more than anything else, the "standard notation" includes a clear, logical presentation in some language (but if that is not English, *I* will have a problem!). It is also important to give some justification for the various design decisions which are usually necessary.

>[...]
>3. Show why your cipher is immune against each of the major attacks
>known in literature. It is not good enough just to say that it is
>secure, you have to show why it is secure against these attacks. This
>requires, of course, that you not only have read the literature, but
>also understand it. Expect this process to take months, and result in
>a large heavily mathematical document. And remember, statistical
>tests are not very meaningful.

That last sentence sounds a lot like statistics-envy. Surely it should read that "statistical tests should not be used to support inappropriate conclusions." But we could say the same thing about mathematics itself.

Even though mathematical cryptography is about 60 years old, it has yet to produce a road map to provable security. This means that all the cryptanalysis and all the arguments about that analysis simply hide the fact that unsuspected and unanalyzed attacks may yet exist.

This does not mean that we do not analyze. But it *does* mean that analysis *cannot* be sufficient, and that makes *testing* important. Testing is often inherently statistical. But many desirable tests are simply *impossible* to perform on a cipher of real size. In my view, that means that no cipher can lay claim to a "thorough" analysis unless it has a scalable architecture, and *is* tested -- necessarily including statistics -- at a tractable size. Not only are statistical

tests *meaningful*, they are all that stands between us and the unknown attack.

Certainly it is going to be very difficult to do a good job fielding *any* cipher system without extensive statistical testing.

>4. Explain why your cipher is better than existing alternatives. It
>makes no sense to look at something new unless it has clear advantages
>over the old stuff. Is it faster on Pentiums? Smaller in hardware?
>What? I have frequently said that, given enough rounds, pretty much
>anything is secure. Your design needs to have significant performance
>advantages. And "it can't be broken" is not an advantage; it's a
>prerequisite.

Note, however, that "performance advantages" include far more than the simple speed of an AES-style cipher box: Large blocks can be an advantage. Dynamically selectable block size can be an advantage. Dynamically variable block size to the byte can be an advantage. Block independence can be an advantage. Self-authentication can be an advantage. There are many advantages which are restricted to particular uses, yet are real advantages in their proper context.

Terry Ritter ritter@io.com http://www.io.com/~ritter/
Crypto Glossary http://www.io.com/~ritter/GLOSSARY.HTM

Subject: Re: Memo to the Amateur Cipher Designer
Date: 22 Oct 1998 01:55:12 GMT
From: olson@umbc.edu (Bryan G. Olson; CMSC (G))
Message-ID: <70m3a0$e2g$1@news.umbc.edu>
References: <362bdbc6.3212829@news.io.com>
Newsgroups: sci.crypt
Lines: 21

Terry Ritter (ritter@io.com) wrote:

Bruce Schneier had written:
: >Most
: >conferences and workshops won't accept designs from unknowns and
: >without extensive analysis. This may seem unfair:

: Not accepting designs from "unknowns" is well beyond just *seeming*
: unfair, it *is* unfair. It is in fact unscientific.

I have to agree with Mr. Ritter on this one. I'll also note that the
major crypto conferences remove the author's name from submission
before they go the referees. The system is based on good faith,
though I have heard referees talk about secure cryptographic protocols
for anonymous review of papers. :)

--Bryan

Subject: Re: Memo to the Amateur Cipher Designer
Date: Thu, 22 Oct 1998 11:37:42 GMT
From: dscott@networkusa.net
Message-ID: <70n5e6kog1@nnrp1.dejanews.com>
References: <70m3a0$e2g$1@news.umbc.edu>
Newsgroups: sci.crypt
Lines: 41

In article <70m3a0$e2g$1@news.umbc.edu>,
olson@umbc.edu (Bryan G. Olson; CMSC (G)) wrote:
> Terry Ritter (ritter@io.com) wrote:
>
> Bruce Schneier had written:
> : >Most
> : >conferences and workshops won't accept designs from unknowns and
> : >without extensive analysis. This may seem unfair:
>
> : Not accepting designs from "unknowns" is well beyond just *seeming*
> : unfair, it *is* unfair. It is in fact unscientific.
>
> I have to agree with Mr. Ritter on this one. I'll also note that the
> major crypto conferences remove the author's name from submission
> before they go the referees. The system is based on good faith,
> though I have heard referees talk about secure cryptographic protocols
> for anonymous review of papers. :)
>
> --Bryan
>

Well Mr Ritter in my view is a much more honest an open person than
Bruce ever will be. At least it is obvious that Ritter works had to
learn and stay abreast of current trends in crypto. SOmething Bruce
is incapable of becasue of his narrow focus and mind set.
I have never never heard of condferences where the author name

is removed. And even if the name is removed I bet any one with have
a brain could tell mine from Bruces and from Mr Ritter since we
all 3 have different writting styles even if we all 3 write about the
exact same subject. Bruces would be acepted even if it left out
key points that Mr Ritter or me may have included since he is the
King of B.S. and he can Pile it Higher and Deeper.

--
http://cryptography.org/cgi-bin/crypto.cgi/Misc/scott19u.zip
ftp search for scott*u.zip in norway

----------== Posted via Deja News, The Discussion Network ==----------
http://www.dejanews.com/ Search, Read, Discuss, or Start Your Own

Originator: markc@news.chiark.greenend.org.uk ([127.0.0.1])

Subject: Re: Memo to the Amateur Cipher Designer
Date: 22 Oct 1998 13:56:30 +0100 (BST)
From: markc@chiark.greenend.org.uk (Mark Carroll)
Message-ID: <+au*FR8In@news.chiark.greenend.org.uk>
References: <70n5e6kog1@nnrp1.dejanews.com>
Newsgroups: sci.crypt
Lines: 47

In article <70n5e6kog1@nnrp1.dejanews.com>, <dscott@networkusa.net> wrote:
(snip)
> Well Mr Ritter in my view is a much more honest an open person than
>Bruce ever will be. At least it is obvious that Ritter works had to
>learn and stay abreast of current trends in crypto. SOmething Bruce
>is incapable of becasue of his narrow focus and mind set.

What current trends in crypto do you think that Bruce isn't abreast of?

> I have never never heard of condferences where the author name
>is removed. And even if the name is removed I bet any one with have

In artificial intelligence (my field) it's very common indeed, even
for the big conferences (as AAAI-98 was). To be honest, I'd be
surprised if it was uncommon in most fields, but I'm open to
correction. What policy do, say, CRYPTO, EUROCRYPT, ASIACRYPT, the
Fast Software Encryption conferences, etc. have? I'd be quite
curious to find out. (-:

>a brain could tell mine from Bruces and from Mr Ritter since we
>all 3 have different writting styles even if we all 3 write about the
>exact same subject. Bruces would be acepted even if it left out

With your Usenet writing style you probably wouldn't get published
anyway, though. The written English in conference proceedings rarely
has copious spelling and grammatical errors; if you were writing a
conference paper, you would no doubt be sensible enough to improve the
English a lot to increase its chances of acceptance. Correct English -
especially in the style of most academic papers - has much less scope
for obvious personal idiosyncrasies (though with rigorous analysis
it's still amazing how personal it turns out to be!).

Certainly, it's sometimes the case that the reviewers guess who the
author(s) might be, but AFAIK it's usually for a tiny minority of
papers, and more from the content than the writing style. (e.g.
X is the only person working on this, and lo and behold here's a
paper about it...)

>key points that Mr Ritter or me may have included since he is the
>King of B.S. and he can Pile it Higher and Deeper.

What interests would the review panel have in choosing Bruce's paper
over yours if yours is so much better? If they start publishing
rubbish, then they'll quickly stop being a major conference (or
jettison any chances of ever becoming one)...

-- Mark

Subject: Re: Memo to the Amateur Cipher Designer
Date: Fri, 23 Oct 1998 00:01:05 GMT
From: dscott@networkusa.net
Message-ID: <70oh01$2o2$1@nnrp1.dejanews.com>
References: <+au*FR8In@news.chiark.greenend.org.uk>
Newsgroups: sci.crypt
Lines: 93

In article <+au*FR8In@news.chiark.greenend.org.uk>,
  markc@chiark.greenend.org.uk (Mark Carroll) wrote:
> In article <70n5e6kog1@nnrp1.dejanews.com>, <dscott@networkusa.net> wrote:
> (snip)
> > Well Mr Ritter in my view is a much more honest an open person than
> >Bruce ever will be. At least it is obvious that Ritter works had to
> >learn and stay abreast of current trends in crypto. SOmething Bruce
> >is incapable of becasue of his narrow focus and mind set.
>
> What current trends in crypto do you think that Bruce isn't abreast of?
>

Well I think the method I use is beyond his tiny limited brain since I don't use the big words he does. I for one fill if you can write in C then that should be enough of an explantion of what is going on in my method. To put it in words gives the wrong impression of it since my use of words will never really say what I want. And then people would enterpit the words differently than I mean.

Bruce only has access to a narrow field of encryption that acidemia uses. I doubt if he understands what Ritter has done either. Even though Ritter is a prolific writter. Bruce might be a phony in only playiing at encryption I have meet many so called Phd where I use to work that lacked any real knowledge of the field they got there degree in. I would've got a Phd in mathematics no sweet but Could not pass all the English stuff that normals could that is why I went into Fields and Waves in Electrical Engineering it had less english crapola.

> > I have never never heard of condferences where the author name
> >is removed. And even if the name is removed I bet any one with have
>
> In artificial intelligence (my field) it's very common indeed, even
> for the big conferences (as AAAI-98 was). To be honest, I'd be
> surprised if it was uncommon in most fields, but I'm open to
> correction. What policy do, say, CRYPTO, EUROCRYPT, ASIACRYPT, the
> Fast Software Encryption conferences, etc. have? I'd be quite
> curious to find out. (-:
>
> >a brain could tell mine from Bruces and from Mr Ritter since we
> >all 3 have different writting styles even if we all 3 write about the
> >exact same subject. Bruces would be acepted even if it left out
>
> With your Usenet writing style you probably wouldn't get published
> anyway, though. The written English in conference proceedings rarely
> has copious spelling and grammatical errors; if you were writing a
> conference paper, you would no doubt be sensible enough to improve the
> English a lot to increase its chances of acceptance. Correct English -
> especially in the style of most academic papers - has much less scope
> for obvious personal idiosyncrasies (though with rigorous analysis
> it's still amazing how personal it turns out to be!).
>

Trust me I can use spell checkers and finally come to correctly spelled words but they wont be the right words anyway so spell checkers don't really add much especially when you think your close to a word and they don't find the one you want or the one they find may be farther than the one you want so you are either stuck with that word or use one you feel is more wrong an a vain attempt to convey your idea. In which your train if thought is lost or broken becasue of the tremendous focus to try to get words in to written form that you can't focus on what you wanted to say in the first place. I hope some day the need for the written word becomes less or that english becomes more like speech and thought.

> Certainly, it's sometimes the case that the reviewers guess who the
> author(s) might be, but AFAIK it's usually for a tiny minority of
> papers, and more from the content than the writing style. (e.g.
> X is the only person working on this, and lo and behold here's a
> paper about it...)
>
> >key points that Mr Ritter or me may have included since he is the
> >King of B.S. and he can Pile it Higher and Deeper.
>
> What interests would the review panel have in choosing Bruce's paper
> over yours if yours is so much better? If they start publishing
> rubbish, then they'll quickly stop being a major conference (or
> jettison any chances of ever becoming one)...
>

Not sure you really want to ask but they may be aready attuned to his narrow closed style of thinking since the reveiwers most like got to there positions in the same way he did and they may not be any more cabable of objective thought than he is.

--
http://cryptography.org/cgi-bin/crypto.cgi/Misc/scott19u.zip
ftp search for scott*u.zip in norway

----------== Posted via Deja News, The Discussion Network ==----------
http://www.dejanews.com/ Search, Read, Discuss, or Start Your Own

---

Subject: Re: Memo to the Amateur Cipher Designer
Date: 22 Oct 1998 19:28:15 GMT
From: aph@cygnus.remove.co.uk (Andrew Haley)
Message-ID: <70o10f$js7$1@korai.cygnus.co.uk>
References: <70n5e6kog1@nnrp1.dejanews.com>
Newsgroups: sci.crypt
Lines: 26

dscott@networkusa.net wrote:
: I have never never heard of condferences where the author name

: is removed.

It's normal procedure.

: And even if the name is removed I bet any one with have a brain
: could tell mine from Bruces and from Mr Ritter

Indeed. They use commas, and can string together two or more
grammatically correct sentences.

Why should anyone be bothered to read what you write if you can't be
bothered to correct any of your mistakes? No publication would put up
with your abysmal English.

Andrew.

Subject: Re: Memo to the Amateur Cipher Designer
Date: 22 Oct 1998 15:58:29 -0500
From: juola@mathcs.duq.edu (Patrick Juola)
Message-ID: <70o69l$291$1@quine.mathcs.duq.edu>
References: <70o10f$js7$1@korai.cygnus.co.uk>
Newsgroups: sci.crypt
Lines: 15

In article <70o10f$js7$1@korai.cygnus.co.uk>,
Andrew Haley <aph@cygnus.remove.co.uk> wrote:
>dscott@networkusa.net wrote:
>:   I have never never heard of condferences where the author name
>: is removed.
>
>It's normal procedure.

Shall I add my voice to the chorus of people pointing out how common
it is? In fact, Mr. Scott, I suspect that part of the reason you
haven't heard of such things is because for most major conferences,
it's expected. I suspect you've never seen a warning label on a
jar of peanut butter stating "warning : will break if dropped," either.

        -kitten

Subject: Re: Memo to the Amateur Cipher Designer
Date: Thu, 22 Oct 1998 23:16:52 GMT
From: ritter@io.com (Terry Ritter)
Message-ID: <362fbcde.3893888@news.io.com>
References: <70o69l$291$1@quine.mathcs.duq.edu>
Newsgroups: sci.crypt
Lines: 35

On 22 Oct 1998 15:58:29 -0500, in <70o69l$291$1@quine.mathcs.duq.edu>,
in sci.crypt juola@mathcs.duq.edu (Patrick Juola) wrote:

>In article <70o10f$js7$1@korai.cygnus.co.uk>,
>Andrew Haley <aph@cygnus.remove.co.uk> wrote:
>>dscott@networkusa.net wrote:
>>:   I have never never heard of condferences where the author name
>>: is removed.
>>
>>It's normal procedure.
>
>Shall I add my voice to the chorus of people pointing out how common
>it is? [...]

Shall I point out that this entire thread is a response to the
original article by Schneier, who wrote:

>[...]
>It's hard to get a cryptographic algorithm published. Most
>conferences and workshops won't accept designs from unknowns and
>without extensive analysis.

Now, presumably Schneier knows something about crypto conferences. He
did *not* say that the practice of removing the author's name for
reviewers was not followed. But he clearly *did* imply that
something prevents "unknowns" from publishing in conferences and
workshops. Maybe he is right. If he is, that is the real issue.

Terry Ritter ritter@io.com http://www.io.com/~ritter/
Crypto Glossary http://www.io.com/~ritter/GLOSSARY.HTM

Subject: Re: Memo to the Amateur Cipher Designer
Date: 23 Oct 1998 09:33:34 -0500
From: juola@mathcs.duq.edu (Patrick Juola)
Message-ID: <70q43u$3tp$1@quine.mathcs.duq.edu>
References: <362fbcde.3893888@news.io.com>
Newsgroups: sci.crypt
Lines: 45

In article <362fbcde.3893888@news.io.com>, Terry Ritter <ritter@io.com> wrote:

[Re: Anonymous review]

>Shall I point out that this entire thread is a response to the
>original article by Schneier, who wrote:
>
>>[...]
>>It's hard to get a cryptographic algorithm published. Most
>>conferences and workshops won't accept designs from unknowns and
>>without extensive analysis.
>
>Now, presumably Schneier knows something about crypto conferences. He
>did *not* say that the practice of removing the author's name for
>reviewers was not followed. But he clearly *did* imply that
>*something* prevents "unknowns" from publishing in conferences and
>workshops. Maybe he is right. If he is, that is the real issue.

Funny how when you eliminate all the "ands" it's very possible to
misinterpret sentences, ya know?

Workshops are generally not reviewed anonymously -- but workshops are
generally not put together as the primary distribution of results;
instead, it's generally a group of people who know each other, or
at least know of each other, getting together to talk shop. In
this sense, they *are*, or can be, closed shops -- which would be
a lot more bothersome if they were taken at all seriously by
professionals.

Conferences are generally reviewed anonymously, especially important
ones. But the standards for conferences are generally much higher --
for instance, most technical conferences require you to submit a
paper, sometimes as much as 10 pages or so, while most workshops only
want an abstract or a one-page summary of what you intend to talk
about. And part of what is expected in the extra nine pages is
a lot more detail about the strengths and weaknesses of what you're
doing.

So if you're a total unknown, you probably won't get workshop invitations.
You can, however, easily get into conferences *if* you can write
a good enough paper -- good enough referring not only to ability to
write decent English but also to the quality of your methodology and
the amount of results.

-kitten

Subject: Re: Memo to the Amateur Cipher Designer
Date: Mon, 26 Oct 1998 04:11:12 GMT
From: schneier@counterpane.com (Bruce Schneier)
Message-ID: <3633f47d.2601979@news.visi.com>
References: <362fbcde.3893888@news.io.com>
Newsgroups: sci.crypt
Lines: 72

On Thu, 22 Oct 1998 23:16:52 GMT, ritter@io.com (Terry Ritter) wrote:

>
>On 22 Oct 1998 15:58:29 -0500, in <70o69l$291$1@quine.mathcs.duq.edu>,
>in sci.crypt juola@mathcs.duq.edu (Patrick Juola) wrote:
>
>>In article <70o10f$js7$1@korai.cygnus.co.uk>,
>>Andrew Haley <aph@cygnus.remove.co.uk> wrote:
>>>dscott@networkusa.net wrote:
>>>: I have never never heard of condferences where the author name
>>>: is removed.
>>>
>>>It's normal procedure.
>>
>>Shall I add my voice to the chorus of people pointing out how common
>>it is? [...]
>
>Shall I point out that this entire thread is a response to the
>original article by Schneier, who wrote:
>
>>[...]
>>It's hard to get a cryptographic algorithm published. Most
>>conferences and workshops won't accept designs from unknowns and
>>without extensive analysis.
>
>Now, presumably Schneier knows something about crypto conferences. He
>did *not* say that the practice of removing the author's name for
>reviewers was not followed. But he clearly *did* imply that
>*something* prevents "unknowns" from publishing in conferences and
>workshops. Maybe he is right. If he is, that is the real issue.

Crypto and Eurocrypt use anonymous refereeing. With a few possible exceptions (I don't know about them all, and Asiacrypt especially) the other crypto conferences keep authors names on the papers during refereeing.

And "hard" is not impossible. Pulling a random (well, pseudorandom) Fast Software Encryption proceedings off my shelf (1997), I see six cipher designs:

> MISTY, by Mitsuru Matsui, the man who intented linear cryptanalysis. It is still unbroken, and I am sorry a variant was not submitted to AES.
>
> ICE, by Matthew Kwan, who has not cryptanalyzed much of anything. Broken in FSE 1998.
>
> TWOPRIME, by Ding, Niemi, Renvall, and Salomaa. Some of these people are good cryptographers, but they are much more mathematicians. I don't think they have ever written a real cryptanalysis paper. TWOPRIME was broken in FSE 1998.
>
> Chameleon, by Ross Anderson and Charalampos Manifavas. Ross has many scalps under his belt. Unbroken.
>
> Square, by Joan Daemen, Lars Knudsen, and Vincent Rijmen, a team that shoud strike fear in the hearts of cipher designers everywhere. Unbroken, and the basis for the AES submission Rijndael.
>
> xmx, by David M'Raihi, David Naccache, Jacques Stern, and Serge Vaudenay. Serge has done some excellent block cipher cryptanalytic work. His design, DFC, has been submitted to AES. Unbroken.

See the pattern?

Bruce

****
Bruce Schneier, President, Counterpane Systems Phone: 612-823-1098
101 E Minnehaha Parkway, Minneapolis, MN 55419 Fax: 612-823-1590
Free crypto newsletter. See: http://www.counterpane.com

Subject: Re: Memo to the Amateur Cipher Designer
Date: Mon, 26 Oct 1998 16:46:05 GMT
From: ritter@io.com (Terry Ritter)
Message-ID: <3634a746.2458264@news.io.com>
References: <3633f47d.2601979@news.visi.com>
Newsgroups: sci.crypt
Lines: 51

On Mon, 26 Oct 1998 04:11:12 GMT, in <3633f47d.2601979@news.visi.com>,
in sci.crypt schneier@counterpane.com (Bruce Schneier) wrote:

>>[...]
>>Now, presumably Schneier knows something about crypto conferences. He
>>did *not* say that the practice of removing the author's name for
>>reviewers was not followed. But he clearly *did* imply that
>>*something* prevents "unknowns" from publishing in conferences and
>>workshops. Maybe he is right. If he is, that is the real issue.
>
>Crypto and Eurocrypt use anonymous refereeing. With a few possible
>exceptions (I don't know about them all, and Asiacrypt especially) the
>other crypto conferences keep authors names on the papers during
>refereeing.
>
>And "hard" is not impossible. Pulling a random (well, pseudorandom)
>Fast Software Encryption proceedings off my shelf (1997), I see six
>cipher designs:
>[...]
>
>See the pattern?

First of all, this is the usual sort of rationalization for treating individuals similarly according to their membership in some sort of despised group. And while clearly unfair and unscientific, it *is* an all-too-American activity.

Next, your argument assumes that science is best served by descriptions of unbreakable cipher designs. But I suggest that they also serve who present new designs of any sort. In fact, it is largely the lack of a broad and robust literature on breaks of all types which makes "the newbie problem" as bad as it is. The process of selecting only good designs for the archival literature leaves us with little description of the bad ones, and less archived reasoning about their weaknesses. I claim we would be better off if every newbie cipher was presented and broken in the literature.

But the original issue wasn't whether limiting crypto conferences to known experts was a reasonable expedient that could be supported by the evidence: The issue instead was whether this occurs. If it does, it is bad science, and if you are participating in this, you are part

of the problem.

See the pattern now?

Terry Ritter ritter@io.com http://www.io.com/~ritter/
Crypto Glossary http://www.io.com/~ritter/GLOSSARY.HTM

Subject: Re: Memo to the Amateur Cipher Designer
Date: Tue, 27 Oct 1998 00:46:37 GMT
From: dscott@networkusa.net
Message-ID: <71355dias1@nnrp1.dejanews.com>
References: <3634a746.2458264@news.io.com>
Newsgroups: sci.crypt
Lines: 68

In article <3634a746.2458264@news.io.com>,
  ritter@io.com (Terry Ritter) wrote:
>
> On Mon, 26 Oct 1998 04:11:12 GMT, in <3633f47d.2601979@news.visi.com>,
> in sci.crypt schneier@counterpane.com (Bruce Schneier) wrote:
>
> >>[...]
> >>Now, presumably Schneier knows something about crypto conferences. He
> >>did *not* say that the practice of removing the author's name for
> >>reviewers was not followed. But he clearly *did* imply that
> >>*something* prevents "unknowns" from publishing in conferences and
> >>workshops. Maybe he is right. If he is, that is the real issue.
> >
> >Crypto and Eurocrypt use anonymous refereeing. With a few possible
> >exceptions (I don't know about them all, and Asiacrypt especially) the
> >other crypto conferences keep authors names on the papers during
> >refereeing.
> >
> >And "hard" is not impossible. Pulling a random (well, pseudorandom)
> >Fast Software Encryption proceedings off my shelf (1997), I see six
> >cipher designs:
> >[...]
> >
> >See the pattern?
>
> First of all, this is the usual sort of rationalization for treating
> individuals similarly according to their membership in some sort of
> despised group. And while clearly unfair and unscientific, it *is* an
> all-too-American activity.
>
> Next, your argument assumes that science is best served by
> descriptions of unbreakable cipher designs. But I suggest that they
> also serve who present new designs of any sort. In fact, it is
> largely the lack of a broad and robust literature on breaks of all
> types which makes "the newbie problem" as bad as it is. The process
> of selecting only good designs for the archival literature leaves us
> with little description of the bad ones, and less archived reasoning
> about their weaknesses. I claim we would be better off if every
> newbie cipher was presented and broken in the literature.
>
> But the original issue wasn't whether limiting crypto conferences to
> known experts was a reasonable expedient that could be supported by
> the evidence: The issue instead was whether this occurs. If it does,
> it is bad science, and if you are participating in this, you are part
> of the problem.
>
> See the pattern now?
>

  Mr RItter I feel that Bruce is one of those self inflated people
incapable of understanding your writting. He is afraid of real
competition so will attempt to put it done with jokes and such
but don't except him to see such an obvious easy pattern in logic
it mat be beyond his brain power.

> ---
> Terry Ritter ritter@io.com http://www.io.com/~ritter/
> Crypto Glossary http://www.io.com/~ritter/GLOSSARY.HTM
>
>

--
http://cryptography.org/cgi-bin/crypto.cgi/Misc/scott19u.zip
ftp search for scott*u.zip in norway

----------== Posted via Deja News, The Discussion Network ==----------
http://www.dejanews.com/ Search, Read, Discuss, or Start Your Own

---

Subject: Re: Memo to the Amateur Cipher Designer
Date: Mon, 26 Oct 1998 18:03:49 GMT
From: jsavard@tenMAPSONeerf.edmonton.ab.ca (John Savard)
Message-ID: <3634b8f5.7503638@news.prosurfr.com>
References: <3633f47d.2601979@news.visi.com>

Newsgroups: sci.crypt
Lines: 16

schneier@counterpane.com (Bruce Schneier) wrote, in part:

> ICE, by Matthew Kwan, who has not cryptanalyzed much of
> anything. Broken in FSE 1998.

But we can be very thankful he published his design. The principle of using a mask to control swapping bits between words is a very useful principle, and can efficiently contribute to a cipher's security.

If ICE hadn't come along, some other cipher designer might have come up with that particular principle, and patented its use in a block cipher.

John Savard
http://members.xoom.com/quadibloc/index.html

Subject: Re: Memo to the Amateur Cipher Designer
Date: Tue, 27 Oct 1998 20:39:45 GMT
From: klockstone@cix.compulink.co.uk ("Keith Lockstone")
Message-ID: <F1I6qA.L1u@cix.compulink.co.uk>
References: <3634b8f5.7503638@news.prosurfr.com>
Newsgroups: sci.crypt
Lines: 89

> But we can be very thankful he published his design. The
> principle of using a mask to control swapping bits between
> words is a very useful principle, and can efficiently
> contribute to a cipher's security.
>
> If ICE hadn't come along, some other cipher designer might have
> come up with that particular principle, and patented its use in
> a block cipher.

This idea has been published before on sci.crypt although its significance probably went unnoticed. It was originally conceived after I read a Friedman publication that described swapping the most significant halves of two indices. There's little new under the Sun!

Keith.

> From: Keith Lockstone <keith.lockstone@gecm.com>
> Newsgroups: sci.crypt
> Subject: Re: New Dynamic Substitution Implications
> Date: 20 Dec 1996 16:48:06 GMT
> Organization: GEC Marconi Radar and Defence Systems Ltd
> To: ritter@io.com
> Lines: 81
>
> In <850598749.26798@dejanews.com>, Terry Ritter said:
> > A better approach would be to use two Dynamic Substitution
> > operations in sequence.
> and:
> > Another approach would be to make a pseudo-random selection
> > among multiple Dynamic Substitution combiners.
>
> A further approach could make use of 4 lookup tables. (Related
> to: Playfair cipher, 4 table ciphers - see William Friedman's
> books on Military Cryptanalysis) This helps to break up patterns
> in the plaintext and the PRNGs.
>
> The basis of this approach is to take pairs of plaintext bytes,
> use them to look up 2 intermediate values, randomly 'splice'
> these to form 2 further intermediate values - which are then used
> to look up the final pair of ciphertext bytes.
>
> All 4 tables are then updated by swapping the used entry with a
> randomly chosen one.
>
> This system has the disadvantage of a random to plaintext ratio
> of 5:2.
>
> Note: if the splicing stage uses 2 random bytes instead of one
> for multiplexing then the system becomes non-reversible - but
> still usable as a mixer for PRNGs.
>
> Keith.
>
> ----------
> #define BYTE unsigned char
> /* tables for encoding (and decoding) */
> BYTE W[256], X[256], Y[256], Z[256];
>
> BYTE p1, p2, /* 2 plaintext input/output bytes */
> c1, c2; /* 2 ciphertext input/output bytes */
>
> /**********/

```
> void encrypt(void)
> {   BYTE r1, r2, r3, r4, r5,                    /* random bytes */
>         a, b, f, g;                    /* intermediate results */
>
>     r1 = getrand(1);              /* get 5 random bytes from 5 */
>
>     r2 = getrand(2);  r3 = getrand(3); /* different generators */
>     r4 = getrand(4);  r5 = getrand(5);
>
>     a = W[p1];         /* plaintext 1 -> intermediate result 1 */
>     b = X[p2];         /* plaintext 2 -> intermediate result 2 */
>
>     f = a & r5 | b & ~r5; /* multiplex intermediate results: 1 */
>     g = b & r5 | a & ~r5; /* multiplex intermediate results: 2 */
>
>     c1 = Y[f];           /* intermediate mix 1 -> ciphertext 1 */
>     c2 = Z[g];           /* intermediate mix 2 -> ciphertext 2 */
>
>     W[p1] = W[r1];  W[r1] = a;               /* update table W */
>     X[p2] = X[r2];  X[r2] = b;               /* update table X */
>     Y[f]  = Y[r3];  Y[r3] = c1;              /* update table Y */
>     Z[g]  = Z[r4];  Z[r4] = c2;              /* update table Z */
> }
```

Subject: Re: Memo to the Amateur Cipher Designer
Date: Fri, 23 Oct 1998 00:24:58 GMT
From: dscott@networkusa.net
Message-ID: <70oicq$3qo$1@nnrp1.dejanews.com>
References: <70o10f$js7$1@korai.cygnus.co.uk>
Newsgroups: sci.crypt
Lines: 52

In article <70o10f$js7$1@korai.cygnus.co.uk>,
  aph@cygnus.remove.co.uk (Andrew Haley) wrote:
> dscott@networkusa.net wrote:
> :   I have never never heard of condferences where the author name
> : is removed.
>
> It's normal procedure.

   Well it is a phony procedure to have the air of respectibility
when in reality it hids nothing. REAL CRYPTO conferences should
have executable program or functions where the input and output
can be analysed and various real testing done on computers.
Since in the real world that is where it has to stand up.
But that might be to difficult and to different for the stuff
onces use to not being creative.

>
> : And even if the name is removed I bet any one with have a brain
> : could tell mine from Bruces and from Mr Ritter
>
> Indeed. They use commas, and can string together two or more
> grammatically correct sentences.
>

 still I meant you could tell Mr Ritter from B.S. by there
styles even though they both like ,s in there stuff.

> Why should anyone be bothered to read what you write if you can't be
> bothered to correct any of your mistakes? No publication would put up
> with your abysmal English.

   Then I guess I can just continue to write the worlds greatest
crypto for the unwashed masses while the one how write can fool
themselves. I don't have to write I can program. May be like
Heavyside some one else will write it in terms that even narrow
minded individuals like Bruce can understand. But I am not
that person.

>
> Andrew.
>
>

--
http://cryptography.org/cgi-bin/crypto.cgi/Misc/scott19u.zip
ftp search for scott*u.zip in norway

----------== Posted via Deja News, The Discussion Network ==----------
http://www.dejanews.com/ Search, Read, Discuss, or Start Your Own

---

Subject: Re: Memo to the Amateur Cipher Designer
Date: Thu, 22 Oct 1998 21:58:09 -0600
From: jgfunj@EnqvbSerrGrknf.pbz (W T Shaw)
Message-ID: <jgfunj-2210982158250001@207.101.116.111>
References: <70oicq$3qo$1@nnrp1.dejanews.com>
Newsgroups: sci.crypt

Lines: 22

In article <70oicq$3qo$1@nnrp1.dejanews.com>, dscott@networkusa.net wrote:

.... even narrow
> minded individuals like Bruce can understand...
>

You are confusing narrow mindedness with focus. I can respect him on that. Certain algorithms are prehaps too demanding of intricate attention, leaving little time for much else.

Each of us is faced with economy of time. These discussions are important, so I spend some effort it them. It is not that I don't have other competing things to do. To keep up seems to take from 20 minutes to several hours per day, but the yield can be much more rapid than running through some sluggish formal procedure. Expanding the fundamental process for introducing and exploring algorithms, prying open the process as far as possible, is more important than any one cryptosystem.
--

Passing a budgit with obscure items is bad; preventing government payment for birth control while authorizing millions for viagra lets us focus on the hard facts of prevalent sexism.

Decrypt with ROT13 to get correct email address.

Subject: Re: Memo to the Amateur Cipher Designer
Date: 23 Oct 1998 09:38:24 -0500
From: juola@mathcs.duq.edu (Patrick Juola)
Message-ID: <70q4d0$3up$1@quine.mathcs.duq.edu>
References: <W%QX1.340$4a.1584242@news20.bellglobal.com>
<70oicq$3qo$1@nnrp1.dejanews.com>
Newsgroups: sci.crypt
Lines: 30

In article <W%QX1.340$4a.1584242@news20.bellglobal.com>,
Sandy Harris <sandy.harris@sympatico.ca> wrote:
>dscott@networkusa.net wrote:
>
>> Then I guess I can just continue to write the worlds greatest
>>crypto for the unwashed masses while the one how write can fool
>>themselves. I don't have to write I can program. . . .
>
>"Besides a mathematical inclination, an exceptionally good mastery of
>"one's native tongue is the most vital asset of a competent programmer.
>
> Edsger W.Dijkstra

It probably helps that Dr. Dijkstra's native tongue is Dutch, where there is a large programming community. It probably also helps that Dr. Dijkstra's command of English is astonishing.

I think he's overstating the case -- I remember a brilliant student I had the pleasure to teach once whos native language was spoken by about two hundred people in a mountain valley in New Guinea or something like that. I suspect that his ability to master English will be a more vital asset for his eventual programming abilities.

But overall, I agree with Dr. Dijkstra's sentiments -- which is itself astonishing, as normally when Dr. Dijkstra states that the sun is shining, my initial reaction is to turn on my headlights. 8-)

-kitten

Subject: Re: Memo to the Amateur Cipher Designer
Date: 23 Oct 1998 15:59:40 GMT
From: aph@cygnus.remove.co.uk (Andrew Haley)
Message-ID: <70q95c$au2$2@korai.cygnus.co.uk>
References: <70q4d0$3up$1@quine.mathcs.duq.edu>
Newsgroups: sci.crypt
Lines: 28

Patrick Juola (juola@mathcs.duq.edu) wrote:
: In article <W%QX1.340$4a.1584242@news20.bellglobal.com>,
: Sandy Harris <sandy.harris@sympatico.ca> wrote:
: >dscott@networkusa.net wrote:
: >
: >"Besides a mathematical inclination, an exceptionally good mastery of
: >"one's native tongue is the most vital asset of a competent programmer.
: >
: > Edsger W.Dijkstra

Dijkstra goes on to explain that in his experience a competent programmer always has such a mastery of his own tongue; in other words, you can gain some idea of the level of a programmer's skill just by listening to them. This tallies with my experience.

: I think he's overstating the case -- I remember a brilliant student
: I had the pleasure to teach once whos native language was spoken

: by about two hundred people in a mountain valley in New Guinea
: or something like that. I suspect that his ability to master
: English will be a more vital asset for his eventual programming
: abilities.

I doubt it; Dijkstra isn't talking about a language skill that someone will actually use to communicate, but is using skill in one's native language as an indicator of linguistic skills in general. After all, one generally thinks in one's native language.

Andrew.

Subject: Re: Memo to the Amateur Cipher Designer
Date: Mon, 26 Oct 1998 13:42:51 +0100
From: Mok-Kong Shen <mok-kong.shen@stud.uni-muenchen.de>
Message-ID: <36346E4B.9720682F@stud.uni-muenchen.de>
References: <70r9tt$i7f$1@nnrp1.dejanews.com>
<3630BE5C.5B765F9B@stud.uni-muenchen.de>
<70q95c$au2$2@korai.cygnus.co.uk>
Newsgroups: sci.crypt
Lines: 26

dscott@networkusa.net wrote:
>

> If you really think in more than one language do you do certain
> things better in one language than the other. Are you using language
> when you play a game like chess. If so do you do better in one than
> the other. Are your politics and views on religion a function some
> what of the language you use at the time your thinking. Just
> wondered.

I can't say that my personal experience generalize. I am however anyway convinced that if one has acquired sufficient proficiency in a foreign language, the difference between a foreign language and one's mother tongue disappears. At that point it is somehow 'uneconomical' to speak or write in one language while thinking partly in another (and then mentally translate before speaking out or writing down) and one tends therefore to work (for convenience) in one single language only. Language proficiency has to be maintained through practice. To my dismay I find my proficiency in my native language (especially in writing) is deteriorating due to lack of practice. Language is neutral to its use. It has no influence on what is expressed, I am convinced, since all natural languages (at least those of the civilized world) are of sufficient expressive power to formulate everything imaginable.

M. K. Shen

Subject: Re: Memo to the Amateur Cipher Designer
Date: Tue, 27 Oct 1998 14:35:46 +0100
From: fungus <spam@egg.chips.and.spam.com>
Message-ID: <3635CC32.F1A238CF@egg.chips.and.spam.com>
References: <36346E4B.9720682F@stud.uni-muenchen.de>
Newsgroups: sci.crypt
Lines: 39

Mok-Kong Shen wrote:
>
> dscott@networkusa.net wrote:
> >
>
> > If you really think in more than one language do you do certain
> > things better in one language than the other.
>
> It has no influence on what is expressed, I am convinced, since
> all natural languages (at least those of the civilized world)
> are of sufficient expressive power to formulate everything
> imaginable.
>

If you actually learn a foreign language you'll find that there are some concepts which have a word in one language but not in another. Sometimes you find yourself arrive at the middle of a sentence wanting to use a word from the other language because no equivalent exists in the language you're speaking.

You'll also notice this in films with subtitles. Sometimes the subtitles are saying a completely different thing than the people on the screen, and, if you think about it, it's very hard to translate directly.

And then there's cultural differences. The concept of swearing (as in "bad language") doesn't really exist in Spain. Over here you'll see Disney films with the characters saying "Oh Shit!", and people say the equivalent of "fuck" all the time on TV chat shows. In the UK you'll have *big* problems to find somebody saying "fuck" on TV. Beverly Hills Cop had all the words changed to "hell" when it was shown....

--

```
&LT;\___/>
/ O O \
\_____/  FTB.
```

Subject: Re: Memo to the Amateur Cipher Designer
Date: Tue, 27 Oct 1998 16:26:40 +0100
From: Mok-Kong Shen <mok-kong.shen@stud.uni-muenchen.de>
Message-ID: <3635E630.DB9FF944@stud.uni-muenchen.de>
References: <3635CC32.F1A238CF@egg.chips.and.spam.com>
Newsgroups: sci.crypt
Lines: 27

fungus wrote:
>

> If you actually learn a foreign language you'll find that there
> are some concepts which have a word in one language but not
> in another. Sometimes you find yourself arrive at the middle
> of a sentence wanting to use a word from the other language
> because no equivalent exists in the language you're speaking.

That's why good translations of master pieces are rare. But I am
not convinced that languages can influence thought or behaviour.
There are always more or less good equivalents. (Though I heard
that in one language one can count up to 5 only.) A language may
be superior in certain expressions but inferior in others. (There
are 'fanatics' who believe that their native languages are the
best.)

> And then there's cultural differences. The concept of swearing
> (as in "bad language") doesn't really exist in Spain.

My respect for Spain. But other languages, including French, long time
the chosen language of the diplomats, are abundant in words
expressing such strong sentiments. I find it difficult to imagine
what happens when two persons get very angry with each other in Spain.

M. K. Shen

Subject: Re: Memo to the Amateur Cipher Designer
Date: 27 Oct 1998 10:35:31 -0500
From: juola@mathcs.duq.edu (Patrick Juola)
Message-ID: <714p83$i54$1@quine.mathcs.duq.edu>
References: <3635CC32.F1A238CF@egg.chips.and.spam.com>
Newsgroups: sci.crypt
Lines: 34

In article <3635CC32.F1A238CF@egg.chips.and.spam.com>,
fungus <spam@egg.chips.and.spam.com> wrote:
>Mok-Kong Shen wrote:
>>
>> dscott@networkusa.net wrote:
>> >
>>
>> > If you really think in more than one language do you do certain
>> > things better in one language than the other.
>>
>> It has no influence on what is expressed, I am convinced, since
>> all natural languages (at least those of the civilized world)
>> are of sufficient expressive power to formulate everything
>> imaginable.
>>
>
>If you actually learn a foreign language you'll find that there
>are some concepts which have a word in one language but not
>in another. Sometimes you find yourself arrive at the middle
>of a sentence wanting to use a word from the other language
>because no equivalent exists in the language you're speaking.

True but irrelevant. Translation doesn't necessarily require
that every word be replaced with an equivalent word, but that
every concept be somehow represented with a word or phrase.

French, for example, has no single word meaning "shallow."

This does NOT, however, mean that the French don't understand
the distinction between deep and shallow water, or even that
they can't talk about it.

        -kitten

Subject: Re: Memo to the Amateur Cipher Designer
Date: Sun, 25 Oct 1998 23:20:50 -0600
From: jgfunj@EnqvbSerrGrknf.pbz (W T Shaw)
Message-ID: <jgfunj-2510982320500001@207.22.198.192>
References: <5LHY1.278$MY4.2154610@news.goodnet.com>
   <jgfunj-2510981033500001@dialup126.itexas.net>

Newsgroups: sci.crypt
Lines: 23

In article <5LHY1.278$MY4.2154610@news.goodnet.com>, "Steve Sampson"
<ssampson@access.usa-site.net> wrote:

> W T Shaw wrote
>
> >I went to Oklahoma yesterday, so on returning to Texas, the contrast with
> >the dimension of past dominated experiences makes me want to confirm my
> >salvation that I recovered by nightfall and express myself in a more
> >linguisting challenging way, at for a little while. Is this prejudice
> >justifed so that I should disregard anything Okie in nature? Perhaps, but
> >I should not bend so easily to such a feeling if I believe that reality is
> >even expressed there.
>
>
> What the hell are you talking about?

Prejudice by language, life style, heritage, anything you want to throw
in. Concentrating on style rather that substance is easy, and wrong.
--

Please excuse if there are multiple postings for my responses...I have no idea where
they come from as I only receive one confirmation for each posting from my
newsserver.

Decrypt with ROT13 to get correct email address.

Subject: Re: Memo to the Amateur Cipher Designer
Date: Mon, 26 Oct 1998 04:02:03 GMT
From: schneier@counterpane.com (Bruce Schneier)
Message-ID: <3633f403.2480424@news.visi.com>
References: <70n5e6kog1@nnrp1.dejanews.com>
Newsgroups: sci.crypt
Lines: 18

On Thu, 22 Oct 1998 11:37:42 GMT, dscott@networkusa.net wrote:
I have never never heard of condferences where the author name
>is removed. And even if the name is removed I bet any one with have
>a brain could tell mine from Bruces and from Mr Ritter since we
>all 3 have different writting styles even if we all 3 write about the
>exact same subject. Bruces would be acepted even if it left out
>key points that Mr Ritter or me may have included since he is the
>King of B.S. and he can Pile it Higher and Deeper.

You can figure out the authors of some papers without the authors'
names, but not all of them. You can easily figure out who is schooled
in the mathematics of cryptography and who isn't.

Bruce
****
Bruce Schneier, President, Counterpane Systems Phone: 612-823-1098
101 E Minnehaha Parkway, Minneapolis, MN 55419 Fax: 612-823-1590
Free crypto newsletter. See: http://www.counterpane.com

Subject: Re: Memo to the Amateur Cipher Designer
Date: Mon, 26 Oct 1998 04:00:47 GMT
From: schneier@counterpane.com (Bruce Schneier)
Message-ID: <3633f3c2.2415094@news.visi.com>
References: <70m3a0$e2g$1@news.umbc.edu>
Newsgroups: sci.crypt
Lines: 28

On 22 Oct 1998 01:55:12 GMT, olson@umbc.edu (Bryan G. Olson; CMSC (G))
wrote:

>Terry Ritter (ritter@io.com) wrote:
>
>Bruce Schneier had written:
>: >Most
>: >conferences and workshops won't accept designs from unknowns and
>: >without extensive analysis. This may seem unfair:
>
>: Not accepting designs from "unknowns" is well beyond just *seeming*
>: unfair, it *is* unfair. It is in fact unscientific.
>
>I have to agree with Mr. Ritter on this one. I'll also note that the
>major crypto conferences remove the author's name from submission
>before they go the referees. The system is based on good faith,
>though I have heard referees talk about secure cryptographic protocols
>for anonymous review of papers. :)

Agreed that it is unfair. But even the conferences that referee
papers anonymously don't publish design papers unless they are REALLY
impressive. Different for the sake of difference just doens't cut it.

Bruce
****
Bruce Schneier, President, Counterpane Systems Phone: 612-823-1098
101 E Minnehaha Parkway, Minneapolis, MN 55419 Fax: 612-823-1590
Free crypto newsletter. See: http://www.counterpane.com

Subject: Re: Memo to the Amateur Cipher Designer
Date: 26 Oct 1998 07:52:04 GMT
From: olson@umbc.edu (Bryan G. Olson; CMSC (G))
Message-ID: <7119n4$7hu$1@news.umbc.edu>
References: <3633f3c2.2415094@news.visi.com>
Newsgroups: sci.crypt
Lines: 21

Bruce Schneier wrote:

: Agreed that it is unfair. But even the conferences that referee
: papers anonymously don't publish design papers unless they are REALLY
: impressive. Different for the sake of difference just doens't cut it.

Oh absolutely. There seems to be a sci.crypt myth that cryptology is primarily concerned with inventing ciphers, and the crypto literature with publishing them. In reality cryptologists are pursuing knowledge within the science of secrecy. The journals and conferences are looking for papers that establish results not previously known.

So here's how to really get a design published in the crypto lit: Find some new and interesting fact, develop a design that incorporates the result, then write a paper that presents both the theorem and the system. I'm still working on mine, but from what I've read, that's how it's usually done.

--Bryan

Subject: Re: Memo to the Amateur Cipher Designer
Date: Mon, 26 Oct 1998 03:59:50 GMT
From: schneier@counterpane.com (Bruce Schneier)
Message-ID: <3633f37a.2343159@news.visi.com>
References: <362bdbc6.3212829@news.io.com>
Newsgroups: sci.crypt
Lines: 9

I invite you to submit a paper, based on your patent #5,727,062 ("Variable Size Block Ciphers") to the 1999 Fast Software Encryption workshop. I believe it will be published.

Bruce
****
Bruce Schneier, President, Counterpane Systems Phone: 612-823-1098
101 E Minnehaha Parkway, Minneapolis, MN 55419 Fax: 612-823-1590
Free crypto newsletter. See: http://www.counterpane.com

Subject: Re: Memo to the Amateur Cipher Designer
Date: Mon, 26 Oct 1998 04:20:14 GMT
From: schneier@counterpane.com (Bruce Schneier)
Message-ID: <3633f6f6.3235025@news.visi.com>
References: <362bdbc6.3212829@news.io.com>
Newsgroups: sci.crypt
Lines: 143

On Tue, 20 Oct 1998 00:40:21 GMT, ritter@io.com (Terry Ritter) wrote:
>In my view, the reason for inventing a new cipher -- like any new
>design -- is to deliver new advantages. If so, it advances the art,
>quite independent of whether the professional chorus agrees.

Security is orthogonal to functionality. A cipher cannot deliver any new advantages until it is considered strong. That's what makes this discipline complicated.

>>What is hard is creating an algorithm that no one else can break, even
>>after years of analysis. And the only way to prove that is to subject
>>the algorithm to years of analysis by the best cryptographers around.
>
>That last sentence is what the profession says, but it is as false,
>misleading, and self-delusional as anything in cryptography: Even
>years of analysis is not proof. It is not close to proof.
>
>Lack of proof of weakness is not proof of strength. Obviously.

Agreed. "Proof" was a bad word choice. You are, of course, correct.

>>It's hard to get a cryptographic algorithm published.
>
>This, of course, is simply false. It is false in the assumption that
>"published" means accepted by some academic journal. And it is also
>more or less false in that most reasonable papers *can* be shopped
>around and placed in some academic journal eventually. There are many
>journals, and each must fill a gaping maw of pages continuously.
>
>It is true, however, that a *good* article is more than a new idea: A
>good article offers *insight* as opposed to mere symbology and
>gobbledygook. If someone provide a good presentation to something
>exciting and new, they won't have much trouble placing it somewhere.

Agreed. Please submit your good ideas to cryptography workshops. FSE and SAC are good places to start.

>>Most
>>conferences and workshops won't accept designs from unknowns and
>>without extensive analysis. This may seem unfair:
>
>Not accepting designs from "unknowns" is well beyond just *seeming*
>unfair, it *is* unfair. It is in fact unscientific. More than that,
>this also has economic consequences.
>
>Any time an academic publication will not look at, accept, and publish
>work based on content -- independent of its source -- that publication
>is spending its academic reputation. Journals exist to serve a larger
>ideal than to simply further the academic careers of authors -- their
>existence depends upon providing the best material to their readers.
>They can't just reject good stuff and publish the chaff without real
>audience consequences.

Agreed. The work is accepted and rejected based on the work, not on the name. If there are errors based on name, it is when a work by a well-known name is refereed less stringently because of who they are. I don't believe the reverse happens anywhere near as often.

>>When I started writing _Applied Cryptography_, I heard the maxim that
>>the only good algorithm designers were people who spent years
>>analyzing existing designs. The maxim made sense, and I believed it.
>
>Then you were fooled. Vernam, a mere engineer in 1919: The
>mechanistic stream cipher, and the basis for the one time pad.

Yes. I believe my point still stands.

>This game of "I'm better than you" is a sickness that infects the
>entire field of cryptography. It makes every discussion a contest,
>every relationship a competition, and a mockery of facts and clear,
>correct reasoning. It works against development in the field, and has
>got to go. Those professionals who are actively cultivating ego cults
>for their own self-gratification are part of the problem.

No. The adversarial game of making and breaking is what makes cryptography cryptography. I design; you break. You design; I break. This is what cryptography is.

>In the anecdote, a better alternative would be for the cryptographer
>to be helpful, to explain the issues, lay out a course of study, and
>thus in a larger sense generally address why the general public has so
>little understanding of this profession. We don't of course, see
>anecdotes about that. Why? See the above paragraph.

I believe we do this. There are excellent courses of study in cryptography that have turned out some excellent cryptographers.

>>1. Describe your cipher using standard notation. This doesn't mean C
>>code. There is established terminology in the literature. Learn it
>>and use it; no one will learn your specialized terminology.
>
>Yes. There are established notations for the design of logic systems,
>and they include both "schematics" and "flow charts" as well as C.
>But more than anything else, the "standard notation" includes a clear,
>logical presentation in some language (but if that is not English, *I*
>will have a problem!). It is also important to give some
>justification for the various design decisions which are usually
>necessary.

I don't mean established notations for the design of logic systems. This is mathematics after all. I mean standard mathematical notation.

>>3. Show why your cipher is immune against each of the major attacks
>>known in literature. It is not good enough just to say that it is
>>secure, you have to show why it is secure against these attacks. This
>>requires, of course, that you not only have read the literature, but
>>also understand it. Expect this process to take months, and result in
>>a large heavily mathematical document. And remember, statistical
>>tests are not very meaningful.
>
>That last sentence sounds a lot like statistics-envy. Surely it
>should read that "statistical tests should not be used to support
>inappropriate conclusions." But we could say the same thing about
>mathematics itself.

No. I stand by my sentence. Statistical tests are not very meaningful. If you saw a cipher design that was accompanied by nothing other than statistical tests of randomness, wouldn't your snake-oil detector go off?

>>4. Explain why your cipher is better than existing alternatives. It
>>makes no sense to look at something new unless it has clear advantages
>>over the old stuff. Is it faster on Pentiums? Smaller in hardware?
>>What? I have frequently said that, given enough rounds, pretty much
>>anything is secure. Your design needs to have significant performance
>>advantages. And "it can't be broken" is not an advantage; it's a
>>prerequisite.
>

>Note, however, that "performance advantages" include far more than the
>simple speed of an AES-style cipher box: Large blocks can be an
>advantage. Dynamically selectable block size can be an advantage.
>Dynamically variable block size to the byte can be an advantage.
>Block independence can be an advantage. Self-authentication can be an
>advantage. There are many advantages which are restricted to
>particular uses, yet are real advantages in their proper context.

Of course.

Bruce
****
Bruce Schneier, President, Counterpane Systems Phone: 612-823-1098
101 E Minnehaha Parkway, Minneapolis, MN 55419 Fax: 612-823-1590
Free crypto newsletter. See: http://www.counterpane.com

Subject: Re: Memo to the Amateur Cipher Designer
Date: Mon, 26 Oct 1998 09:34:59 -0600
From: jgfunj@EnqvbSerrGrknf.pbz (W T Shaw)
Message-ID: <jgfunj-2610980935000001@dialup165.itexas.net>
References: <3633f6f6.3235025@news.visi.com>
Newsgroups: sci.crypt
Lines: 28

In article <3633f6f6.3235025@news.visi.com>, schneier@counterpane.com
(Bruce Schneier) wrote:

>
> I don't mean established notations for the design of logic systems.
> This is mathematics after all. I mean standard mathematical notation.
>
>..... Statistical tests are not very
> meaningful. If you saw a cipher design that was accompanied by
> nothing other than statistical tests of randomness, wouldn't your
> snake-oil detector go off?
>
Statistics can measure more things than randomness. Good logic should be
inclusive rather than exclusive. Calling something snake-oil might mean
that you merely chose not to explore the idea in full, but chose to look
for an excuse to dismiss it, granted that it could be easily applied when
an author will not settle down and converse legitimately about a
particular algorithm; both have nothing to do with whether something is
good or bad.

Randomness would be hard to determine since it includes even things that
don't look random. This is where I start questioning some of the tests
that are touted.
--

Please excuse if there are multiple postings for my responses...I have no idea where
they come from as I only receive one confirmation for each posting from my
newsserver.

Decrypt with ROT13 to get correct email address.

Subject: Re: Memo to the Amateur Cipher Designer
Date: Mon, 26 Oct 1998 18:09:47 GMT
From: jsavard@tenMAPSONeerf.edmonton.ab.ca (John Savard)
Message-ID: <3634ba1b.7797506@news.prosurfr.com>
References: <jgfunj-2610980935000001@dialup165.itexas.net>
Newsgroups: sci.crypt
Lines: 28

jgfunj@EnqvbSerrGrknf.pbz (W T Shaw) wrote, in part:
>In article <3633f6f6.3235025@news.visi.com>, schneier@counterpane.com
>(Bruce Schneier) wrote:

>>..... Statistical tests are not very
>> meaningful. If you saw a cipher design that was accompanied by
>> nothing other than statistical tests of randomness, wouldn't your
>> snake-oil detector go off?

>Randomness would be hard to determine since it includes even things that
>don't look random. This is where I start questioning some of the tests
>that are touted.

And the other way around, things can look nice and random, and even
appear very random to conventional statistical tests, and yet be
vulnerable to the right attack.

For example, I could use the digits of pi as if they were a "one-time
pad", and the result would be beautifully random, but crackable
immediately if someone decided to compare it to pi.

That's all Bruce was saying; statistics aren't enough - although
specialized statistical tests, directly related to the possible forms
of cryptanalysis that a cipher may face, can, of course, be very
applicable.

John Savard
http://members.xoom.com/quadibloc/index.html

Subject: Re: Memo to the Amateur Cipher Designer
Date: Mon, 26 Oct 1998 16:46:15 GMT
From: ritter@io.com (Terry Ritter)
Message-ID: <3634a751.2469260@news.io.com>
References: <3633f6f6.3235025@news.visi.com>
Newsgroups: sci.crypt
Lines: 143

On Mon, 26 Oct 1998 04:20:14 GMT, in <3633f6f6.3235025@news.visi.com>,
in sci.crypt schneier@counterpane.com (Bruce Schneier) wrote:

>On Tue, 20 Oct 1998 00:40:21 GMT, ritter@io.com (Terry Ritter) wrote:
>>In my view, the reason for inventing a new cipher -- like any new
>>design -- is to deliver new advantages. If so, it advances the art,
>>quite independent of whether the professional chorus agrees.
>
>Security is orthogonal to functionality. A cipher cannot deliver any
>new advantages until it is considered strong. That's what makes this
>discipline complicated.

Apparently I have been unable to communicate the issue:

We *never* know that a cipher is strong. Ever.

Now, we might "consider" a cipher strong when all *our* guys have looked at it and found no break. But, quite frankly, the *other* guys have more training, more experience, more resources, more time, and they may even be smarter than our guys. So what does it mean when our guys have anointed a cipher? For the user, it means that even ciphers with a good reputation are *not* guaranteed secure -- and this is the *same* situation they have with unknown ciphers.

I will agree that new cipher designs often have silly errors, and we don't need to be using those ciphers. But if we must greatly reduce the number of ciphers users might have because we don't have the resources to analyze them all, I think we are missing a bet. I claim it is more important to have many different ciphers than to have a few which are "considered strong." Why? Because we *can't* know how strong our ciphers *really* are to the other guy. But we *can* -- guaranteed -- make The Opponent pay dearly to keep up.

>[...]
>>This game of "I'm better than you" is a sickness that infects the
>>entire field of cryptography. It makes every discussion a contest,
>>every relationship a competition, and a mockery of facts and clear,
>>correct reasoning. It works against development in the field, and has
>>got to go. Those professionals who are actively cultivating ego cults
>>for their own self-gratification are part of the problem.
>
>No. The adversarial game of making and breaking is what makes
>cryptography cryptography. I design; you break. You design; I break.
>This is what cryptography is.

I am not referring to legitimate thrust and parry of design and analysis, I am referring to exactly the sort of behavior in your (now deleted) anecdote. I claim:

* The legitimate response to a design is a break.
* The legitimate response to a fixed design is a break.
* The legitimate response to a fixed fixed design is a break.

A humiliating response is never appropriate. And while I am sure we all fail at this goal, we don't all laugh about it, nor do we provide it as an example for others to follow.

Life is tough for cipher analyzers. It must be frustrating when newbies simply do not (no doubt interpreted as "will not") get the point. But *I* am no newbie, and *I* often miss *my* own errors, so I have some sympathy for these guys. I am sure that very few designers quit designing until they are satisfied; almost nobody brings you weak ciphers on purpose.

A big reason that newbies are such a problem is that *we* have failed to communicate cryptography to them. If we had a literature of newbie ciphers and their breaks, we could avoid much of this. But we don't have such a literature specifically because those who complain most about the newbie problem have not allowed that literature to develop. Well, they can't have it both ways.

>[...]
>>>1. Describe your cipher using standard notation. This doesn't mean C
>>>code. There is established terminology in the literature. Learn it
>>>and use it; no one will learn your specialized terminology.
>>
>>Yes. There are established notations for the design of logic systems,
>>and they include both "schematics" and "flow charts" as well as C.
>>But more than anything else, the "standard notation" includes a clear,
>>logical presentation in some language (but if that is not English, *I*
>>will have a problem!). It is also important to give some
>>justification for the various design decisions which are usually

>>necessary.
>
>I don't mean established notations for the design of logic systems.
>This is mathematics after all. I mean standard mathematical notation.

No, cryptography is *not* mathematics.

I suppose that all cryptography can be *described* by mathematics, but that is a far different situation. It is different in the same way that trees can be described by mathematics, but such a description will not contain the essence of "tree-ness," or at least not clearly.

Math descriptions *are* appropriate for essentially mathematical ciphers like number-theoretic designs. But math descriptions are *less* appropriate for logic systems. There is a *reason* most logic designers communicate by schematic: That reason is clarity. Most symmetric ciphers are before all else logic systems, not theorems.

I also note that a math description is hardly a panacea, since 50 years of mathematical cryptography have yet to give us strength. It would be different if we could just take the math description, crank the numbers, and get the answer we want. But we can't. It may be time for a change.

>>>3. Show why your cipher is immune against each of the major attacks
>>>known in literature. It is not good enough just to say that it is
>>>secure, you have to show why it is secure against these attacks. This
>>>requires, of course, that you not only have read the literature, but
>>>also understand it. Expect this process to take months, and result in
>>>a large heavily mathematical document. And remember, statistical
>>>tests are not very meaningful.
>>
>>That last sentence sounds a lot like statistics-envy. Surely it
>>should read that "statistical tests should not be used to support
>>inappropriate conclusions." But we could say the same thing about
>>mathematics itself.
>
>No. I stand by my sentence. Statistical tests are not very
>meaningful. If you saw a cipher design that was accompanied by
>nothing other than statistical tests of randomness, wouldn't your
>snake-oil detector go off?

Not all statistics is frequency testing.

Presumably, one goal in cryptography *ought* to be the coordinated construction of both ciphering structures and statistical tests of those structures which could argue for overall strength. This is a laudable goal, and could be meaningful as hell. But we aren't going to see very much of it if we first discourage everyone from taking that path.

Terry Ritter ritter@io.com http://www.io.com/~ritter/
Crypto Glossary http://www.io.com/~ritter/GLOSSARY.HTM

Subject: Re: Memo to the Amateur Cipher Designer
Date: Tue, 27 Oct 1998 17:37:47 GMT
From: jsavard@tenMAPSONeerf.edmonton.ab.ca (John Savard)
Message-ID: <3635fffc.3625753@news.prosurfr.com>
References: <3634a751.2469260@news.io.com>
Newsgroups: sci.crypt
Lines: 63

ritter@io.com (Terry Ritter) wrote, in part:

>I will agree that new cipher designs often have silly errors, and we
>don't need to be using those ciphers. But if we must greatly reduce
>the number of ciphers users might have because we don't have the
>resources to analyze them all, I think we are missing a bet. I claim
>it is more important to have many different ciphers than to have a few
>which are "considered strong." Why? Because we *can't* know how
>strong our ciphers *really* are to the other guy. But we *can* --
>guaranteed -- make The Opponent pay dearly to keep up.

This is something I basically agree with. Supposing one, or a small handful of ciphers, are so popular that nobody uses anything else: DES, IDEA, Blowfish.

Down the road, despite all the work that has gone into studying them, a weakness that had been overlooked is discovered.

But the recommendations you appear to be making to avoid this danger all seem to have a worse danger: removing the barriers to less credible cipher designers will result in an awful lot of cipher designs with 'silly errors' floating around, with fewer signposts to indicate how to avoid them.

An argument that the barriers are currently too high - that the cryptographic community, as far as symmetric-key systems is concerned, is focused too much on conventional block ciphers to the exclusion of

all else - is something I would be glad to agree with. A radical call to dispense with all barriers, though, doesn't make sense. It makes it look like you think David A. Scott, and others like him, are right; and creating that impression is not going to help your own struggle for a fair hearing.

My own viewpoint is that even if only a limited number of ciphers are analyzed, if these ciphers are representative of a number of basic types, it should be possible to establish groundwork on which essentially trivial variations of these ciphers could be made safely. So that The Opponent doesn't get to attack DES, but DES-alike number 1,394,442.

Symmetric ciphers that don't lengthen the input text don't have much opportunity to leak data and make things worse in a multiple-cipher chain, therefore:

I'd tend to advocate the following as a standard high-security practice: use three ciphers, each from a different tier, on one's secret message. One that is of a type that is very thoroughly analyzed, another one that is different but has recieved some analysis, and something from out in left field - but yet showing some evidence of care in its design, so that it will not be a waste of time.

Even if the less-analyzed cipher does turn out to be weak, one has the example of DESX - certainly XOR by a 64-bit constant is weak - to show that the weak cipher, acting as whitening for the strong one, could still contribute security fully proportionate to (the lesser of) its key size (and that of the stronger cipher).

Making the Opponent work harder is not the same thing as providing the Opponent with an opportunity to get lucky.

John Savard
http://members.xoom.com/quadibloc/index.html

Subject: Re: Memo to the Amateur Cipher Designer
Date: 28 Oct 1998 04:32:35 GMT
From: cbbrowne@news.hex.net (Christopher Browne)
Message-ID: <7166p3$s7p$4@blue.hex.net>
References: <3635fffc.3625753@news.prosurfr.com>
Newsgroups: sci.crypt
Lines: 164

On Tue, 27 Oct 1998 17:37:47 GMT, John Savard
<jsavard@tenMAPSONeerf.edmonton.ab.ca> wrote:
>ritter@io.com (Terry Ritter) wrote, in part:
>
>>I will agree that new cipher designs often have silly errors, and we
>>don't need to be using those ciphers. But if we must greatly reduce
>>the number of ciphers users might have because we don't have the
>>resources to analyze them all, I think we are missing a bet. I claim
>>it is more important to have many different ciphers than to have a few
>>which are "considered strong." Why? Because we *can't* know how
>>strong our ciphers *really* are to the other guy. But we *can* --
>>guaranteed -- make The Opponent pay dearly to keep up.
>
>This is something I basically agree with. Supposing one, or a small
>handful of ciphers, are so popular that nobody uses anything else:
>DES, IDEA, Blowfish.
>
>Down the road, despite all the work that has gone into studying them,
>a weakness that had been overlooked is discovered.

And these are quite valid reasons to encourage as much participation
as possible.

Note that *as possible* does not necessarily imply that *everyone* that
wishes that they could "play the game" gets into the game.

>But the recommendations you appear to be making to avoid this danger
>all seem to have a worse danger: removing the barriers to less
>credible cipher designers will result in an awful lot of cipher
>designs with 'silly errors' floating around, with fewer signposts to
>indicate how to avoid them.

And this is to some extent self-correcting.

In order to be able to *communicate* whether designs are competently
done, it is necessary to have credible *communications* of designs.

That requires having some reasonably expressive common language.

As far as I can tell, the only reasonably universal such language is
that of mathematical notation.

The situation is self-correcting in that those that do not have enough
grasp of common notations such that they can communicate their ideas
will not be heard.

The same is true in various areas of science; there may be some neat
ideas being found out in obscure places relating to many disciplines.

And if those ideas cannot be communicated using the common languages and notations in use, they will not "make it" whether they have merit or not.

The point is that whatever notations get used to describe cryptographic algorithms and systems, they *will* represent isomorphisms to *some* form of mathematical notation.

And if someone is so "independent" of any "established" community that they have notation that is, or nearly is, incomprehensible to the rest of the community, there are several possible causes/effects:

a) Perhaps there is a previously-used notation that nicely represents the algorithm or protocol. "That's a minor variant of Scheider's Feistel cipher.../"

In which case it is preferable for the newcomer to learn the existing notation, so as to be able to fit whatever is new about the cipher into the existing taxonomy.

b) Perhaps the idea really is new and crucial to the community, and should add to the taxonomy.

Which is difficult to determine without having a previous attempt to find isomorphisms that would allow the cipher features to be mapped onto existing notations.

c) The ideas might *not* be crucial or new, and it is thus *not* important for the the community at large to understand the new notation.

There *are* crackpots out there, and lots of them, in virtually any area of scientific endeavor.

In scientific study, it seems to be considered appropriate for people initiating research to try to figure out the common features that new work has with old work.

In the context of crypto research, this implies that a good deal of responsibility for figuring out "where their work fits in" falls to those that come up with new algorithms.

It is all well and good to suggest that those already knowledgeable can help determine taxonomy; Bruce Schneier has done a pretty good job of assisting with this via having written a relatively approachable book that explains many existing ciphers.

>An argument that the barriers are currently too high - that the
>cryptographic community, as far as symmetric-key systems is concerned,
>is focused too much on conventional block ciphers to the exclusion of
>all else - is something I would be glad to agree with.

And they may be focusing that way as:
- Network protocols work with "blocks" of data
- File systems work with "blocks" of data
which all implies that blocks are of fundamental importance.

Further, even a single byte represents a block of 8 bits.

And CPUs are getting increasingly large registers, such that it makes little sense to work with quantities of data much smaller than 64 bits at a time.

In effect, there are many reasons to think blocks are important.

>A radical call
>to dispense with all barriers, though, doesn't make sense. It makes it
>look like you think David A. Scott, and others like him, are right;
>and creating that impression is not going to help your own struggle
>for a fair hearing.

Unfortunately, an algorithm presentation that can't be read due to the use of unconventional notation will be given less attention than one that uses more conventional notation.

And in an area of study where being off by a single bit is expected to make a message into a seemingly random jumble, spelling really does count.

>My own viewpoint is that even if only a limited number of ciphers are
>analyzed, if these ciphers are representative of a number of basic
>types, it should be possible to establish groundwork on which
>essentially trivial variations of these ciphers could be made safely.
>So that The Opponent doesn't get to attack DES, but DES-alike number
>1,394,442.

Evidence in the area of construction of random number generators suggests some contrary evidence; the composition of artificial randomness does not necessarily make things look more random.

Not so incidentally, that suggests further the importance of mathematical analysis and the validity of the use of mathematical notation as the "lingua franca" for cryptography.

>I'd tend to advocate the following as a standard high-security
>practice: use three ciphers, each from a different tier, on one's

>secret message. One that is of a type that is very thoroughly
>analyzed, another one that is different but has recieved some
>analysis, and something from out in left field - but yet showing some
>evidence of care in its design, so that it will not be a waste of
>time.

This isn't an outrageous idea; I would suggest also that it is important to make sure that each "tier" is suitably associated with protocols and (perhaps) appropriate "salting" so that security is not lost via the interfacing of the "tiers." That is, the tiers should be kept as independent as possible so that the evidence found by breaking one level is minimally helpful for attacking other levels.

Otherwise, you may wind up effectively depending on the weakest of the three ciphers...

--
"There are two types of hackers working on Linux: those who can spell, and those who can't. There is a constant, pitched battle between the two camps." --Russ Nelson (Linux Kernel Summary, Ver. 1.1.75 -> 1.1.76)
cbbrowne@ntlug.org- <http//www.hex.net/~cbbrowne/lsf.html>

Subject: Re: Memo to the Amateur Cipher Designer
Date: Wed, 28 Oct 1998 05:21:13 GMT
From: ritter@io.com (Terry Ritter)
Message-ID: <3636a99a.11757150@news.io.com>
References: <3635fffc.3625753@news.prosurfr.com>
Newsgroups: sci.crypt
Lines: 156

On Tue, 27 Oct 1998 17:37:47 GMT, in
<3635fffc.3625753@news.prosurfr.com>, in sci.crypt
jsavard@tenMAPSONeerf.edmonton.ab.ca (John Savard) wrote:

>[...]
>But the recommendations you appear to be making to avoid this danger
>all seem to have a worse danger: removing the barriers to less
>credible cipher designers will result in an awful lot of cipher
>designs with 'silly errors' floating around, with fewer signposts to
>indicate how to avoid them.

I see nothing wrong with ordinary people making their own decisions on cryptography -- or anything else -- based on whatever information they wish to use. If the academics find weakness in particular designs, they can announce that. After some real-world interpretation of those results, people may take steps to move to another cipher.

But this implies that users *have* another cipher, and that it is fairly easy to make a change. Neither of these is likely to be true currently, and I would like to see that change.

>An argument that the barriers are currently too high - that the
>cryptographic community, as far as symmetric-key systems is concerned,
>is focused too much on conventional block ciphers to the exclusion of
>all else - is something I would be glad to agree with.

I do think "the barriers are too high" in the sense that the archival literature tends to avoid what we really want to know. The current ideal article is a cipher with lots of mathematical manipulation -- yet no overall proof of strength -- which rarely if ever supports a reasonable attack.

But I think we generally learn more from attacks than we do from new ciphers. We would thus be better served to have more designs -- including weaker ciphers -- in the archival literature, which would support attacks and so deliver insight not otherwise apparent.

Allowing "weaker" designs into the archival literature would also give us a reasonable way first to handle the unique designs that academics now actively avoid, and second to serve as a roadmap to knowledge for most newbies.

I think a desire to keep the academic literature "pristine" is misguided with respect to cipher designs. Cipher designs cannot be considered "science" in the usual sense anyway, because no new facts are developed and no conclusions proven. This is a design literature, and what we want to know for the future are the failures of the past, in great detail.

>A radical call
>to dispense with all barriers, though, doesn't make sense.

Information security necessarily requires personal commitment. Making individuals (or corporate departments) responsible for using their best judgment on cipher selection seems a very worthwhile tool to get people to pay attention. The cipher itself is almost never the real problem, and paying attention to security can help a lot.

>It makes it

>look like you think David A. Scott, and others like him, are right;
>and creating that impression is not going to help your own struggle
>for a fair hearing.

(I would normally ignore this, but it is a repeat.) If someone is going to judge me by which "side" I seem to be taking, I have little hope that *anything* I *could* present would be received "fairly." The issue is the argument, not who presents it, nor who their acquaintances might be, nor what "side" they are on.

[With respect to "sides," I note that reality is not subject to a popular vote. And I don't think I *have* a "struggle for a fair hearing" -- none of this is about me.]

Many newbies act as they do because they think they are ignored. This is not their delusion, they really *are* ignored. Now, academics may feel that this separates the great unwashed from those of worth, but I think a professor with that point of view should be fired. There really needs to be a better way to help newbies understand where their designs fit in the overall scheme of things. In my view, a "putdown" shows more about the "put-er" than the "put-ee." Experts who cannot explain something simply probably don't really know the subject.

>My own viewpoint is that even if only a limited number of ciphers are
>analyzed, if these ciphers are representative of a number of basic
>types, it should be possible to establish groundwork on which
>essentially trivial variations of these ciphers could be made safely.
>So that The Opponent doesn't get to attack DES, but DES-alike number
>1,394,442.

It would be nice if different cipher versions required significantly different attacks. But since we don't know "the" weakness of a cipher in the first place, it would seem difficult to know which weakness each variation has.

I guess DES-like ciphers might have different tables, and we could index those tables and select among them for each "different" cipher, which might be good enough.

>Symmetric ciphers that don't lengthen the input text don't have much
>opportunity to leak data and make things worse in a multiple-cipher
>chain, therefore:

That is a very good point.

>I'd tend to advocate the following as a standard high-security
>practice: use three ciphers, each from a different tier, on one's
>secret message. One that is of a type that is very thoroughly
>analyzed, another one that is different but has recieved some
>analysis, and something from out in left field - but yet showing some
>evidence of care in its design, so that it will not be a waste of
>time.

And that is another good point, which I intend to adopt. In the past I have not seriously considered multiple ciphering for a production environment, but it may be time to change that. Because the mathematicians among us have not delivered provable strength in practical ciphers, it may be time to argue that multi-ciphering *should* be considered the *expected* operation. We don't need to depend on a single cipher.

Multi-ciphering does seem to require three levels to gain the full strength benefit, and having three different ciphers should be pretty nice. Slow, but nice.

>Even if the less-analyzed cipher does turn out to be weak, one has the
>example of DESX - certainly XOR by a 64-bit constant is weak - to show
>that the weak cipher, acting as whitening for the strong one, could
>still contribute security fully proportionate to (the lesser of) its
>key size (and that of the stronger cipher).
>
>Making the Opponent work harder is not the same thing as providing the
>Opponent with an opportunity to get lucky.

There are several types of "working harder" here. One is the actual deciphering of messages, which I assume you mean.

Another type of "working harder" is the identification, acquisition, and analysis of each cipher variant. And since "many ciphers" means distributing information of value among them, breaking any one means getting only a subset of the information. So with "many ciphers," "attacking" costs more and produces less, an approach which naturally favors the user over the attacker.

Terry Ritter ritter@io.com http://www.io.com/~ritter/
Crypto Glossary http://www.io.com/~ritter/GLOSSARY.HTM

Subject: Re: Memo to the Amateur Cipher Designer
Date: Wed, 28 Oct 1998 12:47:39 -0600
From: jgfunj@EnqvbSerrGrknf.pbz (W T Shaw)
Message-ID: <jgfunj-2810981247390001@dialup136.itexas.net>
References: <3636a99a.11757150@news.io.com>
Newsgroups: sci.crypt
Lines: 21

In article <3636a99a.11757150@news.io.com>, ritter@io.com (Terry Ritter) wrote:

>
> >Symmetric ciphers that don't lengthen the input text don't have much
> >opportunity to leak data and make things worse in a multiple-cipher
> >chain, therefore:
>
> That is a very good point.
>
It's a good point to consider since it is not accurate. It all depends on
what is happening in the algorithms themselves.

You could run the risk of producing some interference pattern in the
combination of algorithms that could produce a poor result, less than what
you want; there are many good examples.
--

Please excuse if there are multiple postings for my responses...I have no idea where
they come from as I only receive one confirmation for each posting from my
newsserver.

Decrypt with ROT13 to get correct email address.

Subject: Re: Memo to the Amateur Cipher Designer
Date: Wed, 28 Oct 1998 19:13:12 GMT
From: ritter@io.com (Terry Ritter)
Message-ID: <36376cba.5685292@news.io.com>
References: <jgfunj-2810981247390001@dialup136.itexas.net>
Newsgroups: sci.crypt
Lines: 30

On Wed, 28 Oct 1998 12:47:39 -0600, in
<jgfunj-2810981247390001@dialup136.itexas.net>, in sci.crypt
jgfunj@EnqvbSerrGrknf.pbz (W T Shaw) wrote:

>In article <3636a99a.11757150@news.io.com>, ritter@io.com (Terry Ritter) wrote:
>
>> >Symmetric ciphers that don't lengthen the input text don't have much
>> >opportunity to leak data and make things worse in a multiple-cipher
>> >chain, therefore:
>>
>> That is a very good point.
>>
>It's a good point to consider since it is not accurate. It all depends on
>what is happening in the algorithms themselves.
>
>You could run the risk of producing some interference pattern in the
>combination of algorithms that could produce a poor result, less than what
>you want; there are many good examples.

While *possible*, in the context of structurally-different ciphers it
is *extremely* unlikely. Indeed, exactly the type of thing we might
be most suspicious of -- encipher, decipher, encipher, using the exact
same cipher -- is widely accepted as Triple DES.

Terry Ritter ritter@io.com http://www.io.com/~ritter/
Crypto Glossary http://www.io.com/~ritter/GLOSSARY.HTM

Subject: Re: Memo to the Amateur Cipher Designer
Date: Thu, 29 Oct 1998 00:07:54 GMT
From: dscott@networkusa.net
Message-ID: <718bkq$t1i$1@nnrp1.dejanews.com>
References: <36376cba.5685292@news.io.com>
Newsgroups: sci.crypt
Lines: 49

In article <36376cba.5685292@news.io.com>,
ritter@io.com (Terry Ritter) wrote:
>
> On Wed, 28 Oct 1998 12:47:39 -0600, in
> <jgfunj-2810981247390001@dialup136.itexas.net>, in sci.crypt
> jgfunj@EnqvbSerrGrknf.pbz (W T Shaw) wrote:
>
> >In article <3636a99a.11757150@news.io.com>, ritter@io.com (Terry Ritter)
wrote:
> >
> >> >Symmetric ciphers that don't lengthen the input text don't have much
> >> >opportunity to leak data and make things worse in a multiple-cipher
> >> >chain, therefore:

> >>
> >> That is a very good point.
> >>
> >It's a good point to consider since it is not accurate. It all depends on
> >what is happening in the algorithms themselves.
> >
> >You could run the risk of producing some interference pattern in the
> >combination of algorithms that could produce a poor result, less than what
> >you want; there are many good examples.
>
> While *possible*, in the context of structurally-different ciphers it
> is *extremely* unlikely. Indeed, exactly the type of thing we might
> be most suspicious of -- encipher, decipher, encipher, using the exact
> same cipher -- is widely accepted as Triple DES.
>

  I don't see why this is not obvious to the socalled experts.
I think they speak highly of Triple DES so as to stay on good terms
with there handlers. It is obvious that mixinf three different types
of ciphers would be better than Triple DES my feelings are that
the NSA can most likely break it easily.
  What do you think Ritter.

> ---
> Terry Ritter ritter@io.com http://www.io.com/~ritter/
> Crypto Glossary http://www.io.com/~ritter/GLOSSARY.HTM
>
>

--
http://cryptography.org/cgi-bin/crypto.cgi/Misc/scott19u.zip
ftp search for scott*u.zip in norway

----------== Posted via Deja News, The Discussion Network ==----------
http://www.dejanews.com/ Search, Read, Discuss, or Start Your Own

---

Subject: Re: Memo to the Amateur Cipher Designer
Date: Thu, 29 Oct 1998 18:40:17 GMT
From: ritter@io.com (Terry Ritter)
Message-ID: <3638b653.7408218@news.io.com>
References: <718bkq$t1i$1@nnrp1.dejanews.com>
Newsgroups: sci.crypt
Lines: 42

On Thu, 29 Oct 1998 00:07:54 GMT, in
<718bkq$t1i$1@nnrp1.dejanews.com>, in sci.crypt dscott@networkusa.net
wrote:

>[...]
> I don't see why this is not obvious to the socalled experts.
>I think they speak highly of Triple DES so as to stay on good terms
>with there handlers. It is obvious that mixinf three different types
>of ciphers would be better than Triple DES my feelings are that
>the NSA can most likely break it easily.
> What do you think Ritter.

I was briefly involved in ANSI X9F3 banking security standards
discussions some years ago, and as I recall there was pressure from
NSA to use only registered ciphers, to avoid Triple DES, and to
prevent multi-ciphering. But maybe that was just disinformation to
make us think Triple DES was strong.

We don't know what NSA can do, and I am not sure it is useful to
speculate. Can they break our strongest ciphers? Well, we really do
desperately need some way to measure or prove cipher strength.
Lacking that, I think large blocks, many ciphers, and multi-ciphering
make a lot of sense, especially if the goal is to achieve
cryptographic levels of assured strength.

But in practice, most of the time, ciphers only need oppose direct
technical attacks which are cheaper than bribery, and that will be a
pretty weak attack. In that sense, weak ciphers may be less of a
problem than having a single fixed cipher that might be cryptanalyzed
once and used to expose everybody.

Since we can't know what NSA can do, I think it can be a waste of time
to worry about it. (Of course, if NSA is doing things a democracy
should not do, that's something else.) I think the danger is less in
what NSA can do, and more in what we refuse to do to help ourselves.

Terry Ritter ritter@io.com http://www.io.com/~ritter/
Crypto Glossary http://www.io.com/~ritter/GLOSSARY.HTM

Subject: Re: Memo to the Amateur Cipher Designer
Date: Thu, 29 Oct 1998 14:10:50 -0500
From: Tim Bass <bass@silkroad.com>
Message-ID: <3638BDBA.7D31E61@silkroad.com>

References: <3638b653.7408218@news.io.com>
Newsgroups: sci.crypt
Lines: 45

> We don't know what NSA can do, and I am not sure it is useful to
> speculate. Can they break our strongest ciphers? Well, we really do
> desperately need some way to measure or prove cipher strength. ...
> Since we can't know what NSA can do, I think it can be a waste of time
> to worry about it.

On a lighter but related note. If all the brainpower used on this
and similar threads (both reading and writing and deleting) were
converted to useful crypto' teamwork, analysis, model development,
etc. NSA or any organization would have little in the way of
'greater abilities and technological advancement'.

From an intellectual perspective, I've read nothing which is
remotely enlightening from this entire thread. The posts with
personal attacks on others and the harsh opinions without
facts are not the words of gentlemen, IMHO. It speaks very
poorly for sci.crypt that those of diverse backgrounds and
opinions cannot discuss relative trivia without resorting
to 'angry hate mail'.

There is no cryptographic algorithm, cipher, or mathematical
implementation which is more important that conducting oneselves
as gentlemen in the face of controversy. My hats off to the many
on sci.crypt who enjoy the pleasure of conducting yourselves
as gentlemen as you are being attacked by the angry and
frustrated minority.

That's all I have to say on this thread. It would be very
pleasurable if we could find a way to harness anger, frustration,
and all the negative energy and create useful, meaningful work
in sci.crypt. Just think of what we could accomplish as
collaborators vis-a-vis antagonists!

My apologies for the raw idealism....

-Tim

--
Tim Bass
Principal Consultant, Systems Engineering
Bass & Associates
Tel: (703) 222-4243
Fax: (703) 222-7320
EMail: bass@silkroad.com.antispam (remove antispam tag)
http://www.silkroad.com/consulting/technical.html

Subject: Re: Memo to the Amateur Cipher Designer
Date: 29 Oct 1998 19:59:12 GMT
From: aph@cygnus.remove.co.uk (Andrew Haley)
Message-ID: <71ahegphd2@korai.cygnus.co.uk>
References: <3638BDBA.7D31E61@silkroad.com>
Newsgroups: sci.crypt
Lines: 10

Tim Bass (bass@silkroad.com) wrote:
: There is no cryptographic algorithm, cipher, or mathematical
: implementation which is more important that conducting oneselves
: as gentlemen in the face of controversy.

How do you expect a female cryptographer feels when told to conduct
herself like a gentleman? That's crude sexism, not raw idealism.
First take the mote from your own eye...

Andrew.

Subject: Re: Memo to the Amateur Cipher Designer
Date: Fri, 30 Oct 1998 05:23:01 GMT
From: "Douglas A. Gwyn" <DAGwyn@null.net>
Message-ID: <36394CD6.333B43BC@null.net>
References: <71ahegphd2@korai.cygnus.co.uk>
Newsgroups: sci.crypt
Lines: 8

Andrew Haley wrote:
> How do you expect a female cryptographer feels when told to conduct
> herself like a gentleman?

He didn't address an individual, he addressed an entire anonymous group.
"Gentlemen" was correct English in that context.
And yes, some of my best friends are women -- but I wouldn't want my
sister to marry one!

Subject: Re: Memo to the Amateur Cipher Designer
Date: Fri, 30 Oct 1998 12:36:42 GMT
From: dscott@networkusa.net
Message-ID: <71cbsqahc1@nnrp1.dejanews.com>

References: <36394CD6.333B43BC@null.net>
Newsgroups: sci.crypt
Lines: 27

In article <36394CD6.333B43BC@null.net>,
  "Douglas A. Gwyn" <DAGwyn@null.net> wrote:
> Andrew Haley wrote:
> > How do you expect a female cryptographer feels when told to conduct
> > herself like a gentleman?
>
> He didn't address an individual, he addressed an entire anonymous group.
> "Gentlemen" was correct English in that context.
> And yes, some of my best friends are women -- but I wouldn't want my
> sister to marry one!
>

   That is a sexist statement if I ever saw one. I think maybe your sister might be better off with a woman. I know I prefer them over men. So it makes sense to me that they might like woman better too.
   Of course of you prefer men like the best of the British crypto people that they the brits only use during war time when the rules are solve the problew or die that is your business. To bad the brits don't have an open mind during peace time. Every body like a little piece know and then.

--
http://cryptography.org/cgi-bin/crypto.cgi/Misc/scott19u.zip
ftp search for scott*u.zip in norway

----------== Posted via Deja News, The Discussion Network ==----------
http://www.dejanews.com/ Search, Read, Discuss, or Start Your Own

---

Subject: Re: Memo to the Amateur Cipher Designer
Date: Fri, 30 Oct 1998 17:07:25 -0500
From: Tim Bass <bass@nospam.silkroad.com>
Message-ID: <363A389D.4988F47B@nospam.silkroad.com>
References: <71d3kqivl1@korai.cygnus.co.uk>
   <36394CD6.333B43BC@null.net>
Newsgroups: sci.crypt
Lines: 52

If an adult was looking at a sandbox full of children and many of the boys were hitting, kicking, spiting, and scratching all the other childern while other boys and girls tried to enjoy playing; then an adult is perfectly correct to use the phrase:

"Please Little Boys, Be Nice, Stop Fighting and Play Together!"

Of course! there always seems to be an angry boy or two in the sandbox who turns their mischief to annoying the one who asked them to be nice :):)

Enlightened adults with sensibility who reads sci.crypt and the personal attacks on many of the good folks during this and other threads, can see whom in the sci.crypt sandbox wants to play together and who wants go throw mud at everyone else.

If "the mudslingers" want to continue to attack others in this sandbox, I suggest they attack them in private email and not in public. And yes, it would be good to behave as "gentlemen". (I have not read any negative comments, harsh speech, nor personal attacks by any of the fairer kinder, calmer, mature, sensible, and more enlightened sex in sci.crypt. woman are far too enlightened, IHMO).

Also, if those whom have been picking on and attacking Mr. Schneier would kindly stop and conduct themselves as gentlemen, it would be much appreciated by many of us. It is really uncalled for and of very poor taste to attack out of malice with the intent to discredit and destroy others.

All restraint from harsh and offensive speech would make sci.crypt a much more positive experience for everyone, IMHO.

- Best Regards,
  Tim

--
   Tim Bass
   Principal Consultant, Systems Engineering
   Bass & Associates
   Tel: (703) 222-4243
   Fax: (703) 222-7320
   EMail: bass@silkroad.com.antispam (remove antispam tag)
   http://www.silkroad.com/consulting/technical.html

Subject: Re: Memo to the Amateur Cipher Designer
Date: Fri, 30 Oct 1998 17:11:35 -0500
From: Tim Bass <bass@nospam.silkroad.com>
Message-ID: <363A3997.93DE323@nospam.silkroad.com>
References: <363A389D.4988F47B@nospam.silkroad.com>
Newsgroups: sci.crypt
Lines: 13

> All restraint from harsh and offensive speech would make sci.crypt
> a much more positive experience for everyone, IMHO.

Obviously, the above sentence should read (my humble apologies):

 Restraint from harsh and offensive speech would make sci.crypt
 a much more positive experience for everyone, IMHO.

Thank you for your cooperation in making sci.crypt a good
experience for everyone!!

-Tim

Subject: Re: Memo to the Amateur Cipher Designer
Date: Sat, 31 Oct 1998 03:34:30 GMT
From: dscott@networkusa.net
Message-ID: <71e0g6$m8p$1@nnrp1.dejanews.com>
References: <363A3997.93DE323@nospam.silkroad.com>
Newsgroups: sci.crypt
Lines: 31

In article <363A3997.93DE323@nospam.silkroad.com>,
  Tim Bass <bass@nospam.silkroad.com> wrote:
> > All restraint from harsh and offensive speech would make sci.crypt
> > a much more positive experience for everyone, IMHO.
>
> Obviously, the above sentence should read (my humble apologies):
>
> Restraint from harsh and offensive speech would make sci.crypt
> a much more positive experience for everyone, IMHO.
>
> Thank you for your cooperation in making sci.crypt a good
> experience for everyone!!
>
> -Tim
>

  Are you for real. Or not I supect a troll if I didn't
know better I would think your last name BASS was a BS clever
attempt since it does sound a little fishy to me. And Bruce
is a Spammer and he laughingly admits it. So what is wrong with
telling him the truth about his self. He is nothing but a
pompous phony.
  At least Ritter has more integrity.

--
http://cryptography.org/cgi-bin/crypto.cgi/Misc/scott19u.zip
ftp search for scott*u.zip in norway

----------== Posted via Deja News, The Discussion Network ==----------
http://www.dejanews.com/ Search, Read, Discuss, or Start Your Own

---

Subject: Re: Memo to the Amateur Cipher Designer
Date: Sat, 31 Oct 1998 06:59:50 GMT
From: "Douglas A. Gwyn" <DAGwyn@null.net>
Message-ID: <363AB508.9E02F0A9@null.net>
References: <71d3kqivl1@korai.cygnus.co.uk>
  <36394CD6.333B43BC@null.net>
Newsgroups: sci.crypt
Lines: 5

Andrew Haley wrote:
> Think about it.

I have thought about it, and trying to change the language
to force one's political views on the world is sickening.

Subject: Re: Memo to the Amateur Cipher Designer
Date: Fri, 30 Oct 1998 01:08:16 GMT
From: dscott@networkusa.net
Message-ID: <71b3i1$p14$1@nnrp1.dejanews.com>
References: <3638b653.7408218@news.io.com>
Newsgroups: sci.crypt
Lines: 56

In article <3638b653.7408218@news.io.com>,
  ritter@io.com (Terry Ritter) wrote:

>
> On Thu, 29 Oct 1998 00:07:54 GMT, in
> <718bkq$t1i$1@nnrp1.dejanews.com>, in sci.crypt dscott@networkusa.net
> wrote:
>
> >[...]
> > I don't see why this is not obvious to the socalled experts.
> >I think they speak highly of Triple DES so as to stay on good terms
> >with there handlers. It is obvious that mixinf three different types
> >of ciphers would be better than Triple DES my feelings are that
> >the NSA can most likely break it easily.
> > What do you think Ritter.
>
> I was briefly involved in ANSI X9F3 banking security standards
> discussions some years ago, and as I recall there was pressure from
> NSA to use only registered ciphers, to avoid Triple DES, and to
> prevent multi-ciphering. But maybe that was just disinformation to
> make us think Triple DES was strong.
>
> We don't know what NSA can do, and I am not sure it is useful to
> speculate. Can they break our strongest ciphers? Well, we really do
> desperately need some way to measure or prove cipher strength.
> Lacking that, I think large blocks, many ciphers, and multi-ciphering
> make a lot of sense, especially if the goal is to achieve
> cryptographic levels of assured strength.
>
> But in practice, most of the time, ciphers only need oppose direct
> technical attacks which are cheaper than bribery, and that will be a
> pretty weak attack. In that sense, weak ciphers may be less of a
> problem than having a single fixed cipher that might be cryptanalyzed
> once and used to expose everybody.
>
> Since we can't know what NSA can do, I think it can be a waste of time
> to worry about it. (Of course, if NSA is doing things a democracy
> should not do, that's something else.) I think the danger is less in
> what NSA can do, and more in what we refuse to do to help ourselves.
>
> ---
> Terry Ritter ritter@io.com http://www.io.com/~ritter/
> Crypto Glossary http://www.io.com/~ritter/GLOSSARY.HTM
>
>

 I liked your answer I just thought I would say so for these
that like to read my posts in case they miss your response.
I have to admit you write better than me. Which of course
is an understatment.

--
http://cryptography.org/cgi-bin/crypto.cgi/Misc/scott19u.zip
ftp search for scott*u.zip in norway

----------== Posted via Deja News, The Discussion Network ==----------
http://www.dejanews.com/ Search, Read, Discuss, or Start Your Own

---

Subject: Re: Memo to the Amateur Cipher Designer
Date: Thu, 29 Oct 1998 10:32:50 -0500
From: Jerry Leichter <leichter@smarts.com>
Message-ID: <36388AA2.3E87@smarts.com>
References: <36376cba.5685292@news.io.com>
Newsgroups: sci.crypt
Lines: 36

| >You could run the risk of producing some interference pattern in the
| >combination of algorithms that could produce a poor result, less
| >than what you want; there are many good examples.
|
| While *possible*, in the context of structurally-different ciphers it
| is *extremely* unlikely.

Not only is it extremely unlikely - it would be a direct indication that
both of the ciphers involved were weaker than expected.

After all, if an attacker has an easier time of attacking E1(E2(X)) than
E2(X), then against a target using E2 he can simply apply E1 himself!
This works for any class of attack, all the way from ciphertext only to
chosen plaintext. (Things are only slightly more subtle for an attack
against E1.)

It *is* essential for this argument that the keys for the two
encryptions be uncorrelated. Then again, you can see that's essential
anyway. As a trivial example, if there were a ciphertext-only attack
against E1, and the key used for E2 could be computed from the one used
from E1, an attack will have no problem with E1(E2(X)).

| Indeed, exactly the type of thing we might
| be most suspicious of -- encipher, decipher, encipher, using the exact
| same cipher -- is widely accepted as Triple DES.

The same argument (with the same restriction) goes through here.
Iterating a cipher is often the start of an attack - it's essential that
there be no (well, almost no) short cycles under iteration. This has
been tested for DES. Interestingly, it doesn't seem to be among the

standard list of things that new ciphers get tested against. I'm
unaware of any general results about, say, Feistel ciphers with certain
kinds of F functions, that guarantee no short cycles. Is this a
potential (if unlikely) vulnerability that's being overlooked?

-- Jerry

Subject: Re: Memo to the Amateur Cipher Designer
Date: Thu, 29 Oct 1998 16:31:12 GMT
From: schneier@counterpane.com (Bruce Schneier)
Message-ID: <36389829.6513924@news.visi.com>
References: <36388AA2.3E87@smarts.com>
Newsgroups: sci.crypt
Lines: 22

On Thu, 29 Oct 1998 10:32:50 -0500, Jerry Leichter
<leichter@smarts.com> wrote:

>| >You could run the risk of producing some interference pattern in the
>| >combination of algorithms that could produce a poor result, less
>| >than what you want; there are many good examples.
>|
>| While *possible*, in the context of structurally-different ciphers it
>| is *extremely* unlikely.
>
>Not only is it extremely unlikely - it would be a direct indication that
>*both* of the ciphers involved were weaker than expected.

Indeed. You cannot prove that a cascade of several ciphers is
stronger than any individual cipher, but is seems reasonable that it
is the case.

Bruce
****
Bruce Schneier, President, Counterpane Systems Phone: 612-823-1098
101 E Minnehaha Parkway, Minneapolis, MN 55419 Fax: 612-823-1590
Free crypto newsletter. See: http://www.counterpane.com

Subject: Re: Memo to the Amateur Cipher Designer
Date: Thu, 29 Oct 1998 15:43:28 -0600
From: jgfunj@EnqvbSerrGrknf.pbz (W T Shaw)
Message-ID: <jgfunj-2910981543280001@dialup105.itexas.net>
References: <36389829.6513924@news.visi.com>
Newsgroups: sci.crypt
Lines: 26

In article <36389829.6513924@news.visi.com>, schneier@counterpane.com
(Bruce Schneier) wrote:

> On Thu, 29 Oct 1998 10:32:50 -0500, Jerry Leichter
> <leichter@smarts.com> wrote:
>
> >| >You could run the risk of producing some interference pattern in the
> >| >combination of algorithms that could produce a poor result, less
> >| >than what you want; there are many good examples.
> >|
> >| While *possible*, in the context of structurally-different ciphers it
> >| is *extremely* unlikely.
> >
> >Not only is it extremely unlikely - it would be a direct indication that
> >*both* of the ciphers involved were weaker than expected.
>
> Indeed. You cannot prove that a cascade of several ciphers is
> stronger than any individual cipher, but is seems reasonable that it
> is the case.
>
Reason requires consideration of details.
--

Please excuse if there are multiple postings for my responses...I have no idea where
they come from as I only receive one confirmation for each posting from my
newsserver.

Decrypt with ROT13 to get correct email address.

Subject: Re: Memo to the Amateur Cipher Designer
Date: Thu, 29 Oct 1998 20:56:54 GMT
From: schneier@counterpane.com (Bruce Schneier)
Message-ID: <3638d619.22372027@news.visi.com>
References: <36388AA2.3E87@smarts.com>
Newsgroups: sci.crypt
Lines: 26

On Thu, 29 Oct 1998 10:32:50 -0500, Jerry Leichter
<leichter@smarts.com> wrote:
>The same argument (with the same restriction) goes through here.
>Iterating a cipher is often the start of an attack - it's essential that
>there be no (well, almost no) short cycles under iteration. This has
>been tested for DES. Interestingly, it doesn't seem to be among the
>standard list of things that new ciphers get tested against. I'm

>unaware of any general results about, say, Feistel ciphers with certain
>kinds of F functions, that guarantee no short cycles. Is this a
>potential (if unlikely) vulnerability that's being overlooked?

I think people are thinking about this, but with long key lengths like
128- and 256 bits, it's hard to make any difinitive statements about
short cycles.

This would be an excellent criterion for someone to analyze at the AES
submissions against. I know of various efforts to look at the AES
submmissions with respect to different attacks, but I have never heard
of anyone looking at the possibilty of short cycles or group
structure.

Bruce
****
Bruce Schneier, President, Counterpane Systems Phone: 612-823-1098
101 E Minnehaha Parkway, Minneapolis, MN 55419 Fax: 612-823-1590
Free crypto newsletter. See: http://www.counterpane.com

Subject: Re: Memo to the Amateur Cipher Designer
Date: Thu, 29 Oct 1998 15:41:00 -0600
From: jgfunj@EnqvbSerrGrknf.pbz (W T Shaw)
Message-ID: <jgfunj-2910981541010001@dialup105.itexas.net>
References: <36388AA2.3E87@smarts.com>
Newsgroups: sci.crypt
Lines: 35

In article <36388AA2.3E87@smarts.com>, Jerry Leichter
<leichter@smarts.com> wrote:
>
> The same argument (with the same restriction) goes through here.
> Iterating a cipher is often the start of an attack - it's essential that
> there be no (well, almost no) short cycles under iteration. This has
> been tested for DES. Interestingly, it doesn't seem to be among the
> standard list of things that new ciphers get tested against. I'm
> unaware of any general results about, say, Feistel ciphers with certain
> kinds of F functions, that guarantee no short cycles. Is this a
> potential (if unlikely) vulnerability that's being overlooked?
>
I seems to be all important. Not testing for this out of fear that you a
weakness would be found seems irresponsible. If you work from certain
premises, in this case that some ciphers are imune to this problem, then
you should want to test to some extent that those ideas actually do hold.

You can only mix a few things in so many ways in a fixed length block
until your ciphertext is identical with one of your previous plaintexts.
Using bigger and more complicated keystructures merely lengthens the
cycle.

Strangely, it does not matter as to which one or several ciphers you use,
the same phenomena must occur as it is axiomatic; only the period will
change, like different structured pseudorandom generators.

To counter the phenomena, I cheat: I change the amount of information in
the block; an interated output can never the the same as a previous
input. Remember, Insanity is doing the same thing over and over again
and expecting a different result; cryptographically, this still holds.
--

Please excuse if there are multiple postings for my responses...I have no idea where
they come from as I only receive one confirmation for each posting from my
newsserver.

Decrypt with ROT13 to get correct email address.

Subject: Re: Memo to the Amateur Cipher Designer
Date: Thu, 29 Oct 1998 21:51:53 GMT
From: ritter@io.com (Terry Ritter)
Message-ID: <3638e349.18919442@news.io.com>
References: <36388AA2.3E87@smarts.com>
Newsgroups: sci.crypt
Lines: 32

On Thu, 29 Oct 1998 10:32:50 -0500, in <36388AA2.3E87@smarts.com>, in
sci.crypt Jerry Leichter <leichter@smarts.com> wrote:

>[...]
>Iterating a cipher is often the start of an attack - it's essential that
>there be no (well, almost no) short cycles under iteration.

I'm not sure I understand this. Presumably "iterating a cipher" means
taking some block, then ciphering it repeatedly until some block value
shows up again, which of course locks us in fixed cycle of states.

A conventional block cipher is a simulated huge Simple Substitution.
So if we look to substitution tables we may see the same issue there.
Certainly Scott has been talking about "single-cycle" tables for a
long time, and I have equally long been questioning what such a
construction would buy. Some attacks are even *defeated* by
multi-cycle tables.

If these "short cycles" are just those which naturally appear in
random permutations, surely a large block is a prescription to make it
unlikely that we could ever find one, or encounter one by chance.

But if the whole purpose here is to make a stream cipher RNG, surely
it would be better to feed the thing from a polynomial counter than to
have it eat its own tail.

Terry Ritter ritter@io.com http://www.io.com/~ritter/
Crypto Glossary http://www.io.com/~ritter/GLOSSARY.HTM

Subject: Re: Memo to the Amateur Cipher Designer
Date: Fri, 30 Oct 1998 02:05:08 GMT
From: dscott@networkusa.net
Message-ID: <71b6sk$t5b$1@nnrp1.dejanews.com>
References: <3638e349.18919442@news.io.com>
Newsgroups: sci.crypt
Lines: 63

In article <3638e349.18919442@news.io.com>,
  ritter@io.com (Terry Ritter) wrote:
>
> On Thu, 29 Oct 1998 10:32:50 -0500, in <36388AA2.3E87@smarts.com>, in
> sci.crypt Jerry Leichter <leichter@smarts.com> wrote:
>
> >[...]
> >Iterating a cipher is often the start of an attack - it's essential that
> >there be no (well, almost no) short cycles under iteration.
>
> I'm not sure I understand this. Presumably "iterating a cipher" means
> taking some block, then ciphering it repeatedly until some block value
> shows up again, which of course locks us in fixed cycle of states.
>
> A conventional block cipher is a simulated huge Simple Substitution.
> So if we look to substitution tables we may see the same issue there.
> Certainly Scott has been talking about "single-cycle" tables for a
> long time, and I have equally long been questioning what such a
> construction would buy. Some attacks are even *defeated* by
> multi-cycle tables.
>

   Yes the Paul Onion attack for a choosen plain test file if
allowed shows that if cycle length known you can taylor an attack
against a pure iterating cipher. If the cycle length not known
one could still use the attack with multipe length choosen files
shorter than the longer one needed for the longer cycle. So if
one was to base it in that it might be best to have 2 or 3 cycles
which is kind of what SKIPJACK used in its S table. However there
are various ways to defeat Maack of X8.zip tried several I think
the round keys was his best and he did not limit his self to a
single cycle. I still feel a single cycle best from an information
point of view and my method of breaking this kind of attack was
to use the Paul routine and for bit rotations on the passes.
If Bruce coughs up his money (follow thread on RE: BOOK RECOM)
then when Joes get it. Someone may win a thousand dollars.
But most millionars are penny pinchers so don't expect to much.

> If these "short cycles" are just those which naturally appear in
> random permutations, surely a large block is a prescription to make it
> unlikely that we could ever find one, or encounter one by chance.
>
> But if the whole purpose here is to make a stream cipher RNG, surely
> it would be better to feed the thing from a polynomial counter than to
> have it eat its own tail.

   But you do like to eat tail don't you Terry?

>
> ---
> Terry Ritter ritter@io.com http://www.io.com/~ritter/
> Crypto Glossary http://www.io.com/~ritter/GLOSSARY.HTM
>
>

--
http://cryptography.org/cgi-bin/crypto.cgi/Misc/scott19u.zip
ftp search for scott*u.zip in norway

----------== Posted via Deja News, The Discussion Network ==----------
http://www.dejanews.com/ Search, Read, Discuss, or Start Your Own

---

Subject: Re: Memo to the Amateur Cipher Designer
Date: Fri, 30 Oct 1998 09:59:32 -0500
From: Jerry Leichter <leichter@smarts.com>
Message-ID: <3639D454.23D5@smarts.com>
References: <3638e349.18919442@news.io.com>

Newsgroups: sci.crypt
Lines: 51

| >[...]
| >Iterating a cipher is often the start of an attack - it's essential
| >that there be no (well, almost no) short cycles under iteration.
|
| I'm not sure I understand this. Presumably "iterating a cipher" means
| taking some block, then ciphering it repeatedly until some block value
| shows up again, which of course locks us in fixed cycle of states.

Assuming an invertible cipher, that cycle must contain the original plaintext. Suppose you knew that fairly short cycles were common. Then a chosen-plaintext attack against a given cipher block X is to feed it back to the encryptor, feed the result, etc. If you're in a short cycle, you'll eventually see X again. The value you saw just before seeing X is the original plaintext.

| A conventional block cipher is a simulated huge Simple Substitution.
| So if we look to substitution tables we may see the same issue there.
| Certainly Scott has been talking about "single-cycle" tables for a
| long time, and I have equally long been questioning what such a
| construction would buy. Some attacks are even *defeated* by
| multi-cycle tables.
|
| If these "short cycles" are just those which naturally appear in
| random permutations, surely a large block is a prescription to make it
| unlikely that we could ever find one, or encounter one by chance.

I can't recall the form of the results on this, but in a truely random subgroup of the permutation group, at least some cycles are certain to be very long. Note that the issue is not the *existence* of short cycles: In a random group, there will be some, but if there aren't many, they can't involve more than a tiny fraction of the elements in the group. (If there is even one cycle of length 1/2 the group, then your chance of picking a key that gives you a short cycle is at most 50%: Half the elements are "already spoken for" by the single long cycle.) What we need to know is that the short cycles - all of whose members correspond to "weak keys" of a sort - amount to only an insignificant fraction of the group. (If there is only one cycle, of course, we know this for certain. Then again, cyclic groups have other limitations for cryptographic purposes.)

Of course, we usually choose subsets of the permutation group that are not actually groups (hence not subgroups). However, the same requirement - not too many short cycles - continues to apply over the group generated by the subset.

| But if the whole purpose here is to make a stream cipher RNG, surely
| it would be better to feed the thing from a polynomial counter than to
| have it eat its own tail.

I don't understand the connection.

-- Jerry

Subject: Re: Memo to the Amateur Cipher Designer
Date: Fri, 30 Oct 1998 18:23:27 GMT
From: ritter@io.com (Terry Ritter)
Message-ID: <363a0414.6595253@news.io.com>
References: <3639D454.23D5@smarts.com>
Newsgroups: sci.crypt
Lines: 85

On Fri, 30 Oct 1998 09:59:32 -0500, in <3639D454.23D5@smarts.com>, in sci.crypt Jerry Leichter <leichter@smarts.com> wrote:

>| >[...]
>| >Iterating a cipher is often the start of an attack - it's essential
>| >that there be no (well, almost no) short cycles under iteration.
>|
>| I'm not sure I understand this. Presumably "iterating a cipher" means
>| taking some block, then ciphering it repeatedly until some block value
>| shows up again, which of course locks us in fixed cycle of states.
>
>Assuming an invertible cipher, that cycle must contain the original
>plaintext.

[This is not a set-up for future attack:] Is it shown that there can be no "lead in" to the short cycles?

>Suppose you knew that fairly short cycles were common. Then
>a chosen-plaintext attack against a given cipher block X is to feed it
>back to the encryptor, feed the result, etc. If you're in a short
>cycle, you'll eventually see X again. The value you saw just before
>seeing X is the original plaintext.

Ah, I see. Chosen-plaintext.

>| A conventional block cipher is a simulated huge Simple Substitution.
>| So if we look to substitution tables we may see the same issue there.
>| Certainly Scott has been talking about "single-cycle" tables for a

>| long time, and I have equally long been questioning what such a
>| construction would buy. Some attacks are even *defeated* by
>| multi-cycle tables.
>|
>| If these "short cycles" are just those which naturally appear in
>| random permutations, surely a large block is a prescription to make it
>| unlikely that we could ever find one, or encounter one by chance.
>
>I can't recall the form of the results on this, but in a truely random
>subgroup of the permutation group, at least some cycles are certain to
>be very long. Note that the issue is not the *existence* of short
>cycles: In a random group, there will be some, but if there aren't
>many, they can't involve more than a tiny fraction of the elements in
>the group. (If there is even one cycle of length 1/2 the group, then
>your chance of picking a key that gives you a short cycle is at most
>50%: Half the elements are "already spoken for" by the single long
>cycle.) What we need to know is that the short cycles - all of whose
>members correspond to "weak keys" of a sort - amount to only an
>insignificant fraction of the group. (If there is only one cycle, of
>course, we know this for certain. Then again, cyclic groups have other
>limitations for cryptographic purposes.)

OK, with the implication being that a random permutation of reasonable size should not have this difficulty.

On the other hand, it seems like we could traverse permutations of modest size and collect cycle-length probability statistics. Then we could measure maximum or minimum cycle lengths, or the distribution itself. Experimental success should improve our confidence that constructions do indeed behave like random permutations. This is probably orthogonal to Boolean function nonlinearity, and may be another reasonable test.

>Of course, we usually choose subsets of the permutation group that are
>not actually groups (hence not subgroups). However, the same
>requirement - not too many short cycles - continues to apply over the
>group generated by the subset.
>
>| But if the whole purpose here is to make a stream cipher RNG, surely
>| it would be better to feed the thing from a polynomial counter than to
>| have it eat its own tail.
>
>I don't understand the connection.

I was just casting about to find the intent of this. Short cycles are a problem in a stream cipher confusion RNG (even for ciphertext only), so maybe the point of not having short cycles was to support that usage. Now I see that was not your point.

Terry Ritter ritter@io.com http://www.io.com/~ritter/
Crypto Glossary http://www.io.com/~ritter/GLOSSARY.HTM

Subject: Re: Memo to the Amateur Cipher Designer
Date: Mon, 02 Nov 1998 11:23:00 -0500
From: Jerry Leichter <leichter@smarts.com>
Message-ID: <363DDC64.412@smarts.com>
References: <363a0414.6595253@news.io.com>
Newsgroups: sci.crypt
Lines: 24

| >Assuming an invertible cipher, that cycle must contain the original
| >plaintext.
|
| [This is not a set-up for future attack:] Is it shown that there
| can be no "lead in" to the short cycles?

That's easy to prove: Suppose there were such a lead in, so that (writing F for the particular encryption under the given key)

```
  A -F-> B -F-> C ...-F-> L -F-> X -F-> X' -F-> Xn+
                                 ^                |
                                 +--------F-------+
```

That is, there's a cycle (X, X', ..., Xn), and there's a lead-in starting A, leading to L, and then L "falls into the cycle" at X.

But then what's F^-1 (F inverse) of X? According to the diagram, both L and Xn map to X under F. But F is supposed to be invertible - it's an encryption algorithm after all, and we'd like to be able to get our plaintext back uniquely! So this diagram is impossible - there cannot be a "lead-in". Rather, L must actually equal Xn (and, working backwards, A must equal one of the Xi's, i.e., must actually *be in* the cycle.)

-- Jerry

Subject: Re: Memo to the Amateur Cipher Designer
Date: Thu, 29 Oct 1998 15:19:03 -0600
From: jgfunj@EnqvbSerrGrknf.pbz (W T Shaw)
Message-ID: <jgfunj-2910981519030001@dialup105.itexas.net>

References: <36376cba.5685292@news.io.com>
Newsgroups: sci.crypt
Lines: 22

In article <36376cba.5685292@news.io.com>, ritter@io.com (Terry Ritter) wrote:

> On Wed, 28 Oct 1998 12:47:39 -0600, in
> <jgfunj-2810981247390001@dialup136.itexas.net>, in sci.crypt
> jgfunj@EnqvbSerrGrknf.pbz (W T Shaw) wrote:
>
> >
> >You could run the risk of producing some interference pattern in the
> >combination of algorithms that could produce a poor result, less than what
> >you want; there are many good examples.
>
> While *possible*, in the context of structurally-different ciphers it
> is *extremely* unlikely. Indeed, exactly the type of thing we might
> be most suspicious of -- encipher, decipher, encipher, using the exact
> same cipher -- is widely accepted as Triple DES.
>
And, we find that the effective keylength is somewhat less than 3 times DES.
--

Please excuse if there are multiple postings for my responses...I have no idea where
they come from as I only receive one confirmation for each posting from my
newsserver.

Decrypt with ROT13 to get correct email address.

Subject: Re: Memo to the Amateur Cipher Designer
Date: Thu, 29 Oct 1998 17:24:13 GMT
From: ssimpson@hertreg.ac.uk
Message-ID: <71a8bt$fm9$1@nnrp1.dejanews.com>
References: <3636a99a.11757150@news.io.com>
Newsgroups: sci.crypt
Lines: 66

In article <3636a99a.11757150@news.io.com>,
  ritter@io.com (Terry Ritter) wrote:

>
> On Tue, 27 Oct 1998 17:37:47 GMT, in
> <3635fffc.3625753@news.prosurfr.com>, in sci.crypt
> jsavard@tenMAPSONeerf.edmonton.ab.ca (John Savard) wrote:
>
> >[...]
> >But the recommendations you appear to be making to avoid this danger
> >all seem to have a worse danger: removing the barriers to less
> >credible cipher designers will result in an awful lot of cipher
> >designs with 'silly errors' floating around, with fewer signposts to
> >indicate how to avoid them.
>
> I see nothing wrong with ordinary people making their own decisions on
> cryptography -- or anything else -- based on whatever information they
> wish to use. If the academics find weakness in particular designs,
> they can announce that. After some real-world interpretation of those
> results, people may take steps to move to another cipher.
>
> But this implies that users *have* another cipher, and that it is
> fairly easy to make a change. Neither of these is likely to be true
> currently, and I would like to see that change.
>
> >An argument that the barriers are currently too high - that the
> >cryptographic community, as far as symmetric-key systems is concerned,
> >is focused too much on conventional block ciphers to the exclusion of
> >all else - is something I would be glad to agree with.
>
> I do think "the barriers are too high" in the sense that the archival
> literature tends to avoid what we really want to know. The current
> ideal article is a cipher with lots of mathematical manipulation --
> yet no overall proof of strength -- which rarely if ever supports a
> reasonable attack.

"Proving" the general security of a block cipher would also prove that P != NP
- something that I don't expect will happen in the near future! If you can
prove it then I'm sure a university or two would like to hear from you :-)

The best we can hope to do is use our complete arsenal of analysis tools to
prove that a cipher is insecure. If it fails to succumb to these tools then
it is not _proven_ to be secure, but it indicates that a degree of faith can
be placed in the cipher.

What other methods would you use to test block ciphers? (e.g. other than all
currently known and published techniques).

If a person presenting a new cipher (e.g. Scott) can't even apply all of the
standard analysis tools then the cipher surely has to be considered "weaker"
than a cipher which passes all of the tests (e.g. TwoFish - which is currently
the object of Scotts hate).

It may be stronger, but empirical evidence suggests not.

Regards,

Sam Simpson
Comms Analyst
-- See http://www.hertreg.ac.uk/ss/ for ScramDisk, a free virtual disk encryption for Windows 95/98. PGP Keys available at the same site.

----------== Posted via Deja News, The Discussion Network ==----------
http://www.dejanews.com/ Search, Read, Discuss, or Start Your Own

---

Subject: Re: Memo to the Amateur Cipher Designer
Date: Fri, 30 Oct 1998 04:19:54 GMT
From: ritter@io.com (Terry Ritter)
Message-ID: <36393e18.1633145@news.io.com>
References: <71a8bt$fm9$1@nnrp1.dejanews.com>
Newsgroups: sci.crypt
Lines: 89

On Thu, 29 Oct 1998 17:24:13 GMT, in
<71a8bt$fm9$1@nnrp1.dejanews.com>, in sci.crypt ssimpson@hertreg.ac.uk
wrote:

>In article <3636a99a.11757150@news.io.com>,
> ritter@io.com (Terry Ritter) wrote:

>>[...]
>> I do think "the barriers are too high" in the sense that the archival
>> literature tends to avoid what we really want to know. The current
>> ideal article is a cipher with lots of mathematical manipulation --
>> yet no overall proof of strength -- which rarely if ever supports a
>> reasonable attack.
>
>"Proving" the general security of a block cipher would also prove that P != NP
>- something that I don't expect will happen in the near future! If you can
>prove it then I'm sure a university or two would like to hear from you :-)

Most block ciphers are not number-theoretic, and I doubt that a proof
of block cipher strength necessarily implies that P <> NP. Indeed, a
block cipher of limited strength may be all we will ever need, if we
could only prove that it has *enough* strength.

>The best we can hope to do is use our complete arsenal of analysis tools to
>prove that a cipher is insecure. If it fails to succumb to these tools then
>it is not _proven_ to be secure, but it indicates that a degree of faith can
>be placed in the cipher.

Concluding that a cipher which has not been shown weak is therefore
strong is surely incorrect reasoning. So the cipher may be weak. And
if the cipher *is* weak, we surely would be fools to have faith in it,
no matter how much analysis was done previously.

The evidence we get from analysis simply does not support a conclusion
that a worked-on cipher is more deserving of "faith" than a new
cipher. I think we make this logical leap because the result is
comforting, because it seems to reward the effort spent in analysis,
and because we seem to have little choice. But that does not make the
reasoning right, or the conclusion correct.

Indeed, for all we know, there may *be* no strong cipher. And that
would mean that the partitioning of ciphers into "weak" and "strong"
is an irrelevant illusion.

>What other methods would you use to test block ciphers? (e.g. other than all
>currently known and published techniques).

We should test everything we can, and then understand that everything
we have not tested is in an unknown state. If we can't test it, we
can't control it. And in ciphers, we cannot test strength.

>If a person presenting a new cipher (e.g. Scott) can't even apply all of the
>standard analysis tools then the cipher surely has to be considered "weaker"
>than a cipher which passes all of the tests (e.g. TwoFish - which is currently
>the object of Scotts hate).

That is the reasoning which is generally applied, but that reasoning
is false. This is precisely the point I have been addressing: When
we don't know, we *really* don't know. And we can't draw correct
conclusions from not knowing.

The only thing we can prove with analysis is a limit on strength; that
the real strength could not exceed the effort of a given break. But
it does not say that there is not a weaker break, somewhere, if we
only could see deeper, or understand more. Analysis cannot say that
the analyzed cipher is stronger than an unanalyzed cipher.

In fact, the regrettable situation with regard to the academic
literature implies that analysis results generally will not be
published if no attack is found. This means that any cipher we call

"analyzed" will have a break of some sort. Do we really trust a
cipher which has a known break more than one which does not?

>It may be stronger, but empirical evidence suggests not.

There is no such "evidence." There is no support for a correct
conclusion one way or the other. When we choose rumor and innuendo to
support a conclusion, we have no reason to expect a correct result.

Terry Ritter ritter@io.com http://www.io.com/~ritter/
Crypto Glossary http://www.io.com/~ritter/GLOSSARY.HTM

Subject: Re: Memo to the Amateur Cipher Designer
Date: Fri, 30 Oct 1998 09:51:46 GMT
From: ssimpson@hertreg.ac.uk
Message-ID: <71c27h$ue1$1@nnrp1.dejanews.com>
References: <36393e18.1633145@news.io.com>
Newsgroups: sci.crypt
Lines: 104

In article <36393e18.1633145@news.io.com>,
  ritter@io.com (Terry Ritter) wrote:
>
> On Thu, 29 Oct 1998 17:24:13 GMT, in
> <71a8bt$fm9$1@nnrp1.dejanews.com>, in sci.crypt ssimpson@hertreg.ac.uk
> wrote:
>
> >In article <3636a99a.11757150@news.io.com>,
> > ritter@io.com (Terry Ritter) wrote:
>
> >>[...]
> >> I do think "the barriers are too high" in the sense that the archival
> >> literature tends to avoid what we really want to know. The current
> >> ideal article is a cipher with lots of mathematical manipulation --
> >> yet no overall proof of strength -- which rarely if ever supports a
> >> reasonable attack.
> >
> >"Proving" the general security of a block cipher would also prove that P !=
NP
> >- something that I don't expect will happen in the near future! If you can
> >prove it then I'm sure a university or two would like to hear from you :-)
>
> Most block ciphers are not number-theoretic, and I doubt that a proof
> of block cipher strength necessarily implies that P <> NP. Indeed, a
> block cipher of limited strength may be all we will ever need, if we
> could only prove that it has *enough* strength.
>

I was quoting pg 52 (right column) of the paper "Twofish: A 128-bit Block
Cipher" by Schneier, Kelsey, Whiting, Wagner, Hall & Ferguson.

> >The best we can hope to do is use our complete arsenal of analysis tools to
> >prove that a cipher is insecure. If it fails to succumb to these tools then
> >it is not _proven_ to be secure, but it indicates that a degree of faith can
> >be placed in the cipher.
>
> Concluding that a cipher which has not been shown weak is therefore
> strong is surely incorrect reasoning. So the cipher may be weak. And
> if the cipher *is* weak, we surely would be fools to have faith in it,
> no matter how much analysis was done previously.

But we have to have faith in one (or possibly more) block ciphers. Rather
than pick this cipher at "random" it is surely better to pick the a block
cipher that has been subjected to and resisted all known attacks.

For example I would pick Blowfish over ICE. Wouldn't you?

> Indeed, for all we know, there may *be* no strong cipher. And that
> would mean that the partitioning of ciphers into "weak" and "strong"
> is an irrelevant illusion.

Quite. It may be true that no strong ciphers are strong or weak. But at the
moment we can certainly point our fingers at ciphers that *are* weak and
others that are relatively secure. (e.g. ICE *is* weak and should not be
used. Blowfish has not been shown to be weak and as such can be trusted).

> >What other methods would you use to test block ciphers? (e.g. other than all
> >currently known and published techniques).
>
> We should test everything we can, and then understand that everything
> we have not tested is in an unknown state. If we can't test it, we
> can't control it. And in ciphers, we cannot test strength.

Indeed. But more faith has to be placed in a block cipher that has undergone
all tests (and passed) rather than a cipher that has not been tested
thoroughly.

If you disagree on this point then I think we'll just have to "agree to
disagree".

For now, I'm not putting untested ciphers into ScramDisk :-)

>
> >If a person presenting a new cipher (e.g. Scott) can't even apply all of the
> >standard analysis tools then the cipher surely has to be considered "weaker"
> >than a cipher which passes all of the tests (e.g. TwoFish - which is
currently
> >the object of Scotts hate).
>
<SNIP>
>
> The only thing we can prove with analysis is a limit on strength; that
> the real strength could not exceed the effort of a given break. But
> it does not say that there is not a weaker break, somewhere, if we
> only could see deeper, or understand more. Analysis cannot say that
> the analyzed cipher is stronger than an unanalyzed cipher.
>

No. But they can say that the analysed cipher (that has passed the tests) has
more credibility.

<SNIP>

Regards,

Sam Simpson
Comms Analyst
-- See http://www.hertreg.ac.uk/ss/ for ScramDisk, a free virtual disk
encryption for Windows 95/98. PGP Keys available at the same site.

----------== Posted via Deja News, The Discussion Network ==----------
http://www.dejanews.com/ Search, Read, Discuss, or Start Your Own

---

Subject: Re: Memo to the Amateur Cipher Designer
Date: Fri, 30 Oct 1998 19:02:45 GMT
From: ritter@io.com (Terry Ritter)
Message-ID: <363a0d51.8960237@news.io.com>
References: <71c27h$ue1$1@nnrp1.dejanews.com>
Newsgroups: sci.crypt
Lines: 127

On Fri, 30 Oct 1998 09:51:46 GMT, in
<71c27h$ue1$1@nnrp1.dejanews.com>, in sci.crypt ssimpson@hertreg.ac.uk
wrote:

>>[...]
>> Concluding that a cipher which has not been shown weak is therefore
>> strong is surely incorrect reasoning. So the cipher may be weak. And
>> if the cipher *is* weak, we surely would be fools to have faith in it,
>> no matter how much analysis was done previously.
>
>But we have to have faith in one (or possibly more) block ciphers.

I guess faith is about the only thing we *can* have. But that's
religion, not science. We may use a cipher, but we *cannot* trust it.

>Rather
>than pick this cipher at "random" it is surely better to pick the a block
>cipher that has been subjected to and resisted all known attacks.

Frankly, I have come to believe that it may be more important to use a
multiplicity of ciphers -- accepting their possible weaknesses -- than
to use a single cipher -- and accepting its possible weakness.

I think what we gain from an analysis, and from wide use, is that
successful attacks are unlikely from people like us, or those who
tried to break the cipher. So if the purpose of the cipher is to
prevent people like us from getting in, there is some reason to think
that analysis has given us that assurance.

But if we intend to stop people who are better trained, better funded,
and who have more experience, time, and equipment than us -- and who
may even be smarter than we are -- our attempts at analysis tell us
nothing at all. They may in fact delude us into the belief that
nobody can be better at doing what we try to do than we are.

This is why we need to innovate and use protocols which allow us to
accept cipher weakness, yet continue to get the job done.

>For example I would pick Blowfish over ICE. Wouldn't you?

Not if picking Blowfish means that I have no access to other ciphers.

Since any cipher may have weakness, a widely used cipher is asking for
weakness to be found. And using any cipher for a large amount of data
is asking for weakness to be exploited.

The exploitation of our data is the risk. How can we possibly "put

all our eggs in one basket" when we know that it is *impossible* to
"watch that basket"?

>> Indeed, for all we know, there may *be* no strong cipher. And that
>> would mean that the partitioning of ciphers into "weak" and "strong"
>> is an irrelevant illusion.
>
>Quite. It may be true that no strong ciphers are strong or weak. But at the
>moment we can certainly point our fingers at ciphers that *are* weak and
>others that are relatively secure. (e.g. ICE *is* weak and should not be
>used. Blowfish has not been shown to be weak and as such can be trusted).

Fine. But not being shown weak still does not mean we can trust it.
How can we possible trust something to be strong which admittedly may
be weak?

>> >What other methods would you use to test block ciphers? (e.g. other than all
>> >currently known and published techniques).
>>
>> We should test everything we can, and then understand that everything
>> we have not tested is in an unknown state. If we can't test it, we
>> can't control it. And in ciphers, we cannot test strength.
>
>Indeed. But more faith has to be placed in a block cipher that has undergone
>all tests (and passed) rather than a cipher that has not been tested
>thoroughly.
>
>If you disagree on this point then I think we'll just have to "agree to
>disagree".

I do indeed disagree. One cannot gain faith about untested things by
testing other things. There is no reason to expect that the outcome
of future tests will be like past successes. If this were generally
true, we would never need complex tests.

>For now, I'm not putting untested ciphers into ScramDisk :-)

I did say "test everything we can." But I think the idea of selecting
some subset of ciphers for inclusion in a program should gradually
fade away. (All but one of these will be unused at any particular
time anyway.)

>>[...]
>> The only thing we can prove with analysis is a limit on strength; that
>> the real strength could not exceed the effort of a given break. But
>> it does not say that there is not a weaker break, somewhere, if we
>> only could see deeper, or understand more. Analysis cannot say that
>> the analyzed cipher is stronger than an unanalyzed cipher.
>>
>
>No. But they can say that the analysed cipher (that has passed the tests) has
>more credibility.

There is a lot to be said for experience. By experiencing many of the
ways things can go wrong, one can take steps to avoid those problems.
And this extends far beyond the cipher into the design and
implementation of the cipher system. I suspect that problems in the
cipher system are likely to be more easily exploited than any
publicly-described cipher.

When we first install a normal program, we may not trust it. After we
use it for a while, and it has not failed, we may develop trust. And
certainly we can trust a cipher system in the same way when we talk
about data not being lost or scrambled. But we can't trust strength,
because use does not stress that, and its failure is not reported to
us. We have no idea whether the program really is delivering strength
or not, so we cannot develop a trust of strength.

Terry Ritter ritter@io.com http://www.io.com/~ritter/
Crypto Glossary http://www.io.com/~ritter/GLOSSARY.HTM

Subject: Re: Memo to the Amateur Cipher Designer
Date: Fri, 30 Oct 1998 05:15:28 GMT
From: "Douglas A. Gwyn" <DAGwyn@null.net>
Message-ID: <36394B11.2DDFF3CE@null.net>
References: <71a8bt$fm9$1@nnrp1.dejanews.com>
Newsgroups: sci.crypt
Lines: 7

ssimpson@hertreg.ac.uk wrote:
> "Proving" the general security of a block cipher would also prove that P != NP

I wonder on what basis you could make that claim.
In other words, I don't think that's right -- I can exhibit the design
for a block cipher that is demonstrably secure according to the rules of
the game, although it wouldn't be *practical*.

Subject: Re: Memo to the Amateur Cipher Designer
Date: Fri, 30 Oct 1998 05:22:15 GMT
From: schneier@counterpane.com (Bruce Schneier)
Message-ID: <36394cbf.326485@news.visi.com>
References: <36394B11.2DDFF3CE@null.net>
Newsgroups: sci.crypt
Lines: 19

On Fri, 30 Oct 1998 05:15:28 GMT, "Douglas A. Gwyn" <DAGwyn@null.net>
wrote:
>ssimpson@hertreg.ac.uk wrote:
>> "Proving" the general security of a block cipher would also prove that P != NP
>
>I wonder on what basis you could make that claim.
>In other words, I don't think that's right -- I can exhibit the design
>for a block cipher that is demonstrably secure according to the rules of
>the game, although it wouldn't be *practical*.

While it is certainly possible to, in theory, give a proof of security
that does not also prove that P != NP, most formulations of such a
proof--which, of course, does not exist--hinge on proving P != NP.

Bruce
****
Bruce Schneier, President, Counterpane Systems Phone: 612-823-1098
101 E Minnehaha Parkway, Minneapolis, MN 55419 Fax: 612-823-1590
Free crypto newsletter. See: http://www.counterpane.com

Subject: Re: Memo to the Amateur Cipher Designer
Date: Sat, 31 Oct 1998 06:34:34 GMT
From: "Douglas A. Gwyn" <DAGwyn@null.net>
Message-ID: <363AAF1C.AF37BF7F@null.net>
References: <36394cbf.326485@news.visi.com>
Newsgroups: sci.crypt
Lines: 15

Bruce Schneier wrote:
> On Fri, 30 Oct 1998 05:15:28 GMT, "Douglas A. Gwyn" <DAGwyn@null.net>
> wrote:
> >ssimpson@hertreg.ac.uk wrote:
> >> "Proving" the general security of a block cipher would also prove that P != NP
> >I wonder on what basis you could make that claim.
> >In other words, I don't think that's right -- I can exhibit the design
> >for a block cipher that is demonstrably secure according to the rules of
> >the game, although it wouldn't be *practical*.
> While it is certainly possible to, in theory, give a proof of security
> that does not also prove that P != NP, most formulations of such a
> proof--which, of course, does not exist--hinge on proving P != NP.

This is getting weirder -- I still would like a reference.
I don't think *any* block ciphers have anything to do with P?=NP.

Subject: Re: Memo to the Amateur Cipher Designer
Date: Mon, 02 Nov 1998 20:07:17 GMT
From: jsavard@tenMAPSONeerf.edmonton.ab.ca (John Savard)
Message-ID: <363e0fff.13076835@news.prosurfr.com>
References: <363AAF1C.AF37BF7F@null.net>
Newsgroups: sci.crypt
Lines: 19

"Douglas A. Gwyn" <DAGwyn@null.net> wrote, in part:

>This is getting weirder -- I still would like a reference.
>I don't think *any* block ciphers have anything to do with P?=NP.

Not in the simple way that some public-key systems do.

But someone did note that he had converted DES into a gigantic Boolean
expression ... in hopes that it could be, at least partly, inverted. I
think inverting logic equations does touch on P versus NP.

Essentially, if a proof that P=NP is interpreted as indicating there
are no mathematical problems that get really intractable to solve,
compared to the effort required to verify the solution, then that
would seem to affect everything - even if the application to
secret-key ciphers would still be awkwards.

John Savard
http://members.xoom.com/quadibloc/index.html

Subject: Re: Memo to the Amateur Cipher Designer
Date: 2 Nov 1998 15:31:14 -0500
From: juola@mathcs.duq.edu (Patrick Juola)
Message-ID: <71l4qirmv1@quine.mathcs.duq.edu>
References: <363e0fff.13076835@news.prosurfr.com>
Newsgroups: sci.crypt
Lines: 40

In article <363e0fff.13076835@news.prosurfr.com>,

John Savard <jsavard@tenMAPSONeerf.edmonton.ab.ca> wrote:
>"Douglas A. Gwyn" <DAGwyn@null.net> wrote, in part:
>
>>This is getting weirder -- I still would like a reference.
>>I don't think *any* block ciphers have anything to do with P?=NP.
>
>Not in the simple way that some public-key systems do.
>
>But someone did note that he had converted DES into a gigantic Boolean
>expression ... in hopes that it could be, at least partly, inverted. I
>think inverting logic equations does touch on P versus NP.

Only if the size of the problem varies.

In the case of any *particular* block cypher, with any *particular* key-space and any *particular* block size, &c, then the problem size is probably fixed (and P/NP is indeed a red herring). So proving that P == NP probably wouldn't affect the solution of DES much.

However, a lot of cyphers are in fact cypher schemes with variable size. An obvious example is creating an LFSR-based stream cypher, where the difficulty of recreating the stream can be related to the size of the LFSR-state (and hence to the size of the secret key).

If I were to develop a clever trick with N-bit LFSRs and prove that to recover the internal state or decrypt the cyphertext *required* at least 2^N operations, then I would indeed have proven that P != NP.

A similar example, this one involving a block cypher, would be if I used N-bit RSA, but kept both factors secret (and part of the key). This, of course, means that I lose the advantages of public-key encryption. On the other hand, it also means that I have a symmetric algorithm with a variable problem size.

If I could then *prove* that the only way to decrypt my messages required exponential time (in the size of the RSA keys), then I would, again, have proven P < NP.

-kitten

Subject: Re: Memo to the Amateur Cipher Designer
Date: Tue, 03 Nov 1998 00:21:06 GMT
From: "Douglas A. Gwyn" <DAGwyn@null.net>
Message-ID: <363E4C10.C05E0898@null.net>
References: <363e0fff.13076835@news.prosurfr.com>
Newsgroups: sci.crypt
Lines: 9

John Savard wrote:
> But someone did note that he had converted DES into a gigantic Boolean
> expression ... in hopes that it could be, at least partly, inverted. I
> think inverting logic equations does touch on P versus NP.

But the issue is not whether there is an *effective algorithm* for inverting *every* system of equations, which might bear on P?=NP. The statement was that proof of security of *any particular example* of a block cipher system would imply P=NP. That's what I doubt.

Subject: Re: Memo to the Amateur Cipher Designer
Date: Mon, 02 Nov 1998 20:49:04 -0500
From: Nicol So <nobody@no.spam.please>
Message-ID: <363E6110.9D3FF87@no.spam.please>
References: <363E4C10.C05E0898@null.net>
Newsgroups: sci.crypt
Lines: 48

Douglas A. Gwyn wrote:

> But the issue is not whether there is an *effective algorithm* for
> inverting *every* system of equations, which might bear on P?=NP.
> The statement was that proof of security of *any particular example*
> of a block cipher system would imply P=NP. That's what I doubt.

Whether a proof of security of a block cipher has anything to do with the question of P?=NP depends on how you formalize the notion of security. Theoreticians (I mean theoretical computer scientists) like to define security in terms of asymptotic properties. For the purpose of this discussion, ignore whether such definitions properly capture the notion of security for practical ciphers. There are different formalizations of security, some of which are not applicable to (deterministic) block ciphers. For example, no deterministic ciphers can be semantically secure. A probably more applicable notion of security for block ciphers is that of superpseudorandom permutation generator, as introduced in a 1986 paper by Luby and Rackoff.

To use the definition, a block cipher is modeled as a (uniform) family of polynomial-time computable permutations, indexed by a security parameter k. The intuition behind superpseudorandom permutation generator is that a block cipher is secure if it passes off as a (length-preserving) permutation on {0,1}^k, and no (non-uniform) polynomial-time algorithm can distinguish them. The (non-uniform) algorithm here takes two oracles: a "normal" one computing a function f, and an "inverse" one computing the inverse of f.

"Distinguish", as used in the definition, means "distinguishes with a non-negligible advantage". (If you make random guesses, you already have a 50% success rate of saying whether a given permutation is truly random or just pseudorandom). "Negligible" here has the conventional meaning of "converging to 0 faster than the inverse of any polynomial k^c (where c>0), as the security parameter k tends to infinity.

So much for the background definitions. Now consider a particular block cipher (modeled as a family of ciphers of variable block sizes and key lengths). In non-deterministic polynomial time, an algorithm can determine, with good success probability, whether a given pair of encryption/decryption functions could have been an instance of our block cipher (with some appropriate key). This can be done by by guessing a key and perform a number of trial encryptions and decryptions.

If P=NP, the above computation can also be performed in polynomial-time. That means, for any given cipher, there is always a polynomial-time algorithm that can "distinguish" the cipher from permutations chosen uniformly at random from the appropriate space. And therefore, no deterministic block cipher can be secure under such a definition of security.

Subject: Re: Memo to the Amateur Cipher Designer
Date: Tue, 03 Nov 1998 20:25:23 -0500
From: Nicol So <nobody@no.spam.please>
Message-ID: <363FAD03.75C29461@no.spam.please>
References: <363E6110.9D3FF87@no.spam.please>
Newsgroups: sci.crypt
Lines: 14

Nicol So wrote:

> ...
> The intuition behind superpseudorandom permutation generator is that a block
> cipher is secure if it passes off as a (length-preserving) permutation on
> {0,1}^k, and no (non-uniform) polynomial-time algorithm can distinguish
> them. ...

When I wrote "permutation on {0,1}^k", I meant to say "permutation on {0,1}^k chosen uniformly at random". Despite the accidental omission, the intended meaning should be obvious from the ensuing text.

Nicol

Subject: Re: Memo to the Amateur Cipher Designer
Date: 3 Nov 1998 09:32:34 -0500
From: juola@mathcs.duq.edu (Patrick Juola)
Message-ID: <71n462sgp1@quine.mathcs.duq.edu>
References: <363E4C10.C05E0898@null.net>
Newsgroups: sci.crypt
Lines: 35

In article <363E4C10.C05E0898@null.net>,
Douglas A. Gwyn <DAGwyn@null.net> wrote:
>John Savard wrote:
>> But someone did note that he had converted DES into a gigantic Boolean
>> expression ... in hopes that it could be, at least partly, inverted. I
>> think inverting logic equations does touch on P versus NP.
>
>But the issue is not whether there is an *effective algorithm* for
>inverting *every* system of equations, which might bear on P?=NP.
>The statement was that proof of security of *any particular example*
>of a block cipher system would imply P=NP. That's what I doubt.

The demonstration of a particular category of equations, such that

a) "size" is meaningful
and
b) to determine whether or not they are satisfiable provably
requires exponential time

would indeed prove that P < NP. The reason, of course, is that the general case encompasses the specific case, and no general equation-solver could solve these particular problems faster than the provable bound -- so the general equation solver would also require exponential time.

However, this is a one-way implication. If I could prove that DES (or any particular sub-class of the general problem) *were* solvable in polynomial time, this would NOT prove that P == NP.

-kitten

Subject: Re: Memo to the Amateur Cipher Designer
Date: Tue, 03 Nov 1998 17:00:14 GMT
From: bobs@rsa.com
Message-ID: <71ncqv$j7d$1@nnrp1.dejanews.com>
References: <363e0fff.13076835@news.prosurfr.com>
Newsgroups: sci.crypt
Lines: 17

In article <363e0fff.13076835@news.prosurfr.com>,
  jsavard@tenMAPSONeerf.edmonton.ab.ca (John Savard) wrote:
> "Douglas A. Gwyn" <DAGwyn@null.net> wrote, in part:

> Essentially, if a proof that P=NP is interpreted as indicating there
> are no mathematical problems that get really intractable to solve,
> compared to the effort required to verify the solution, then that
> would seem to affect everything - even if the application to
> secret-key ciphers would still be awkwards.

Such an interpretation would be grossly wrong.

It is well known that problems exist that are HARDER than any problems
in NP. See Garey & Johnson, for example.

----------== Posted via Deja News, The Discussion Network ==----------
http://www.dejanews.com/ Search, Read, Discuss, or Start Your Own

---

Subject: Re: Memo to the Amateur Cipher Designer
Date: Tue, 03 Nov 1998 23:05:41 GMT
From: "Douglas A. Gwyn" <DAGwyn@null.net>
Message-ID: <363F8BE2.7469BAAF@null.net>
References: <71ncqv$j7d$1@nnrp1.dejanews.com>
Newsgroups: sci.crypt
Lines: 17

bobs@rsa.com wrote:
> In article <363e0fff.13076835@news.prosurfr.com>,
> jsavard@tenMAPSONeerf.edmonton.ab.ca (John Savard) wrote:
> > "Douglas A. Gwyn" <DAGwyn@null.net> wrote, in part:
>
> > Essentially, if a proof that P=NP is interpreted as indicating there
> > are no mathematical problems that get really intractable to solve,
> > compared to the effort required to verify the solution, then that
> > would seem to affect everything - even if the application to
> > secret-key ciphers would still be awkwards.
>
> Such an interpretation would be grossly wrong. ...

Please, check the attributions before posting.
You posted "Douglas A. Gwyn wrote:" followed by
text that I certainly did not write.
(Presumably it was written by John Savard.)

Subject: Re: Memo to the Amateur Cipher Designer
Date: Fri, 30 Oct 1998 10:38:15 GMT
From: ssimpson@hertreg.ac.uk
Message-ID: <71c4ul$1fl$1@nnrp1.dejanews.com>
References: <36394B11.2DDFF3CE@null.net>
Newsgroups: sci.crypt
Lines: 30

In article <36394B11.2DDFF3CE@null.net>,
  "Douglas A. Gwyn" <DAGwyn@null.net> wrote:
> ssimpson@hertreg.ac.uk wrote:
> > "Proving" the general security of a block cipher would also prove that P !=
NP
>
> I wonder on what basis you could make that claim.
> In other words, I don't think that's right -- I can exhibit the design
> for a block cipher that is demonstrably secure according to the rules of
> the game, although it wouldn't be *practical*.

I was quoting pg 52 (right-hand column) of the paper "Twofish: A 128-bit Block
Cipher" by Schneier, Kelsey, Whiting, Wagner, Hall & Ferguson.

I would be interested in your views on this. Are Schneier et al wrong? Have
I miss something? Did I (gasp!) take the quote out of context?

I am honestly interested in an answer if the statement was wrong as I'm
relatively new to encryption e.g. less than 7 years, so am still learning.

Thanks,

Sam Simpson
Comms Analyst
-- See http://www.hertreg.ac.uk/ss/ for ScramDisk, a free virtual disk
encryption for Windows 95/98. PGP Keys available at the same site.

----------== Posted via Deja News, The Discussion Network ==----------
http://www.dejanews.com/ Search, Read, Discuss, or Start Your Own

---

Subject: Re: Memo to the Amateur Cipher Designer
Date: Fri, 30 Oct 1998 12:51:38 GMT
From: sandy.harris@sympatico.ca (Sandy Harris)
Message-ID: <uDi_1.603$Gh4.1162471@news21.bellglobal.com>
References: <71c4ul$1fl$1@nnrp1.dejanews.com>
Newsgroups: sci.crypt
Lines: 25

ssimpson@hertreg.ac.uk wrote:

> "Douglas A. Gwyn" <DAGwyn@null.net> wrote:
>> ssimpson@hertreg.ac.uk wrote:
>> > "Proving" the general security of a block cipher would also prove that P !=
>NP
>>
>> I wonder on what basis you could make that claim.

Encryption/decryption with known key is presumably not worse than
polynomial in keylength or the cipher's wildly impractical.

If "proving the security" of the cipher means showing that no attack
is better than brute force, i.e. all possible attacks are exponential in
keylength, & if this applies for any keylength, then QED.

Methinks this argument is hopelessly flawed because the keylength
in most ciphers cannot vary beyond a certain range & the whole
P/NP distinction depends on reasoning for "in the limit" & "for
sufficiently large N", so it cannot reasonably be applied.

Of course if you consider an iterated block cipher with independent
round keys & a variable # of rounds, then the total key can be
arbitrarily large, so perhaps the argument is salvagable for such
ciphers.

Subject: Re: Memo to the Amateur Cipher Designer
Date: Fri, 30 Oct 1998 16:29:15 GMT
From: bobs@rsa.com
Message-ID: <71cpgquqd1@nnrp1.dejanews.com>
References: <uDi_1.603$Gh4.1162471@news21.bellglobal.com>
Newsgroups: sci.crypt
Lines: 29

In article <uDi_1.603$Gh4.1162471@news21.bellglobal.com>,
  sandy.harris@sympatico.ca (Sandy Harris) wrote:
> ssimpson@hertreg.ac.uk wrote:
>
> > "Douglas A. Gwyn" <DAGwyn@null.net> wrote:
> >> ssimpson@hertreg.ac.uk wrote:
> >> > "Proving" the general security of a block cipher would also prove that P !=
> >NP
> >>
> >> I wonder on what basis you could make that claim.
>
> Encryption/decryption with known key is presumably not worse than
> polynomial in keylength or the cipher's wildly impractical.
>
> If "proving the security" of the cipher means showing that no attack
> is better than brute force, i.e. all possible attacks are exponential in
> keylength, & if this applies for any keylength, then QED.

No! No! No!

Proving that only brute force could work would NOT, repeat NOT
prove P != NP *** unless **** you first proved that breaking the key
was an NP-Complete problem.

Merely showing that breaking the key takes exponential time is NOT
equivalent to proving it is NP-Complete.

----------== Posted via Deja News, The Discussion Network ==----------
http://www.dejanews.com/ Search, Read, Discuss, or Start Your Own

---

Subject: Re: Memo to the Amateur Cipher Designer
Date: 30 Oct 1998 14:44:46 -0500
From: juola@mathcs.duq.edu (Patrick Juola)
Message-ID: <71d4ve$p8o$1@quine.mathcs.duq.edu>
References: <71cpgquqd1@nnrp1.dejanews.com>
Newsgroups: sci.crypt
Lines: 37

In article <71cpgquqd1@nnrp1.dejanews.com>, <bobs@rsa.com> wrote:
>In article <uDi_1.603$Gh4.1162471@news21.bellglobal.com>,
> sandy.harris@sympatico.ca (Sandy Harris) wrote:
>> ssimpson@hertreg.ac.uk wrote:
>>
>> > "Douglas A. Gwyn" <DAGwyn@null.net> wrote:
>> >> ssimpson@hertreg.ac.uk wrote:
>> >> > "Proving" the general security of a block cipher would also prove that P !=
>> >NP
>> >>
>> >> I wonder on what basis you could make that claim.

>>
>> Encryption/decryption with known key is presumably not worse than
>> polynomial in keylength or the cipher's wildly impractical.
>>
>> If "proving the security" of the cipher means showing that no attack
>> is better than brute force, i.e. all possible attacks are exponential in
>> keylength, & if this applies for any keylength, then QED.
>
>No! No! No!
>
>Proving that only brute force could work would NOT, repeat NOT
>prove P != NP *** unless **** you first proved that breaking the key
>was an NP-Complete problem.
>
>Merely showing that breaking the key takes exponential time is NOT
>equivalent to proving it is NP-Complete.

Nope. Showing that breaking the key takes *provably* exponential time
would suffice to show that P != NP. If there exists a subset S of NP
such that P < S <= NP, that proves P < NP. (S, in this case, is the
class of problems to which this cypher belongs.)

And of course, the problem is "obviously" in NP because you can verify
a correct solution in polynomial time.

-kitten

Subject: Re: Memo to the Amateur Cipher Designer
Date: Fri, 30 Oct 1998 21:49:07 GMT
From: phr@netcom.com (Paul Rubin)
Message-ID: <phrF1ntxv.4ys@netcom.com>
References: <71cpgquqd1@nnrp1.dejanews.com>
Newsgroups: sci.crypt
Lines: 24

In article <71cpgquqd1@nnrp1.dejanews.com>, <bobs@rsa.com> wrote:
>Proving that only brute force could work would NOT, repeat NOT
>prove P != NP *** unless **** you first proved that breaking the key
>was an NP-Complete problem.
>
>Merely showing that breaking the key takes exponential time is NOT
>equivalent to proving it is NP-Complete.

Bob, are you sure of this? If the statement came from someone less
knowledgeable than you, I'd have shrugged it off as wrong rather
than paying attention.

If brute force works, then cryptanalizing the cipher is in NP. Once
you have guessed the key by brute force, you can validate the guess by
showing it properly encrypts the known plaintext to the known
ciphertext.

If you can prove that *only* brute force works, the cipher is not in P.
Brute force means exponential search through the keyspace. It is true
that cryptanalyzing the cipher may not be NP-hard, but it is not in P.

If something is in NP but not in P, it follows that P != NP.

Did I miss something??!!

Subject: Re: Memo to the Amateur Cipher Designer
Date: Mon, 02 Nov 1998 20:12:24 GMT
From: jsavard@tenMAPSONeerf.edmonton.ab.ca (John Savard)
Message-ID: <363e112c.13377666@news.prosurfr.com>
References: <phrF1ntxv.4ys@netcom.com>
Newsgroups: sci.crypt
Lines: 22

phr@netcom.com (Paul Rubin) wrote, in part:

>If you can prove that *only* brute force works, the cipher is not in P.
>Brute force means exponential search through the keyspace. It is true
>that cryptanalyzing the cipher may not be NP-hard, but it is not in P.

I think the idea is that while a _proof_ that only brute force works
would indeed catapult cryptanalyzing it out of P, in general the fact
that only brute force is known at present (which some people might
take for a proof) certainly doesn't have anything to do with P versus
NP.

And secret-key designs can easily be made much too messy for anything
to be proven about them...

So, at present I think you're right and they're not expressing
themselves clearly (they're right too about what they're trying to
say). If I'm wrong, and there is a reason why the P=NP question
doesn't apply, even in theory, that would be interesting.

John Savard
http://members.xoom.com/quadibloc/index.html

Subject: Re: Memo to the Amateur Cipher Designer
Date: Tue, 03 Nov 1998 11:21:42 -0700
From: Shawn Willden <shawn@willden.org>
Message-ID: <363F49B6.8CE95B8@willden.org>
References: <363e112c.13377666@news.prosurfr.com>
Newsgroups: sci.crypt
Lines: 34

John Savard wrote:

> So, at present I think you're right and they're not expressing
> themselves clearly (they're right too about what they're trying to
> say). If I'm wrong, and there is a reason why the P=NP question
> doesn't apply, even in theory, that would be interesting.

Let me see if I can lay this out clearly and thoroughly enough that someone can point out the flaw in the reasoning (Douglas Gwyn? Bob Silverman?).

P is the set of all problems that are solvable in polynomial time.

NP is the set of all problems for which candidate solutions can be tested in polynomial time.

P is a subset of NP. To see this, choose a problem p in P and a candidate solution c, run the polynomial-time algorithm to solve p which yields a solution s, then test if c=s.

So, P=NP iff NP is a subset of P. Therefore, to show P!=NP, it is sufficient to show that there exists a problem p s.t. p is an element of NP but p is not an element of P.

Let p be a problem whose candidate solutions can be tested in polynomial time (p is an element of NP) but which requires (provably) that an exponentially growing solution space be brute force searched to find a solution. This implies that p is not an element of P, which shows that P != NP.

What's wrong with that argument? Or is there something wrong with my definitions of P and NP?

Shawn.

Subject: Re: Memo to the Amateur Cipher Designer
Date: Mon, 02 Nov 1998 20:16:46 GMT
From: jsavard@tenMAPSONeerf.edmonton.ab.ca (John Savard)
Message-ID: <363e1282.13719546@news.prosurfr.com>
References: <71cpgquqd1@nnrp1.dejanews.com>
Newsgroups: sci.crypt
Lines: 23

bobs@rsa.com wrote, in part:

>Proving that only brute force could work would NOT, repeat NOT
>prove P != NP   *** unless ****   you first proved that breaking the key
>was an NP-Complete problem.

>Merely showing that breaking the key takes exponential time is NOT
>equivalent to proving it is NP-Complete.

Proving that brute force was not necessary would not prove P=NP, unless you proved that breaking the key was NP-complete, since there are problems that aren't known to be in P, but aren't known to be NP-complete either, like factoring.

But the converse _is_ valid, unless my memory is very faulty: NP-complete problems are supposed to be the hardest kind of scalable problems; thus, if some problem was shown not to be in P, even if that problem was _not_ NP complete, that would only mean that the NP-complete problems were as hard, or even harder, and therefore not in P either.

John Savard
http://members.xoom.com/quadibloc/index.html

Subject: Re: Memo to the Amateur Cipher Designer
Date: Sat, 31 Oct 1998 06:47:39 GMT
From: "Douglas A. Gwyn" <DAGwyn@null.net>
Message-ID: <363AB22D.D9725C8@null.net>
References: <uDi_1.603$Gh4.1162471@news21.bellglobal.com>
Newsgroups: sci.crypt
Lines: 11

Sandy Harris wrote:
> Encryption/decryption with known key is presumably not worse than
> polynomial in keylength or the cipher's wildly impractical.

Granted.

> If "proving the security" of the cipher means showing that no attack
> is better than brute force, i.e. all possible attacks are exponential in

> keylength, & if this applies for any keylength, then QED.

No, that's not even close to a proof of: <given cipher secure> => P!=NP.

Subject: Re: Memo to the Amateur Cipher Designer
Date: 2 Nov 1998 07:31:30 GMT
From: olson@umbc.edu (Bryan G. Olson; CMSC (G))
Message-ID: <71jn4i$510$1@news.umbc.edu>
References: <363AB22D.D9725C8@null.net>
Newsgroups: sci.crypt
Lines: 67

Douglas A. Gwyn (DAGwyn@null.net) wrote:
: Sandy Harris wrote:
: > Encryption/decryption with known key is presumably not worse than
: > polynomial in keylength or the cipher's wildly impractical.

: Granted.

: > If "proving the security" of the cipher means showing that no attack
: > is better than brute force, i.e. all possible attacks are exponential in
: > keylength, & if this applies for any keylength, then QED.

: No, that's not even close to a proof of: <given cipher secure> => P!=NP.

Hmmm, I see it as kind of close. From the assumption that
(en/de)cryption is polytime and attack is exponential in key
length, we must be talking about a cipher with a variable size
and arbitrarily large key. Now let's define encryption as

f(K,M) = C

Where K is the key, M is the plaintext, and C is the ciphertext.
Lets say K is an integer, since any key can be coded in that
form.

Now we define language L as the set of all strings of the form:

[m,c,x]

Where there exists some key k such that:

k < x and f(k,m) = c.

L is in NP. We assumed (en/de)cryption is polytime, so for any
string in L, there is a short certifier - namely a value for k
that satisfies the "such that: ...CRYPHTML.HTM" above.

L is not in P. I'll prove this by contradiction. If L is in P,
then given plaintext m and ciphertext c, I can recover k such
that f(m,k)=p using the following procedure:

Test the strings [m,c,2], [m,c,4], [m,c,8]...
for membership in L until I find the first that is in L.

Now I have two strings [m,c,x1] and [m,c,x2] where x1<x2,
[m,c,x1] is not in L, and [m,c,x2] is in L. I now test
whether [m,c,floor((x1+x2)/2)] is in L. Either way, I can
divide in half the interval containing the lowest x' such
that [m,c,x'] is in L. I repeat this procedure to form a
binary search for x', and when I find it I return k=x'.

The procedure takes time proportional to the time to test whether
a string is in L times the length of K. Since we assumed L in
in P, this gives me a sub-exponential break of the cipher, and
since our premise says such a break doesn't exist, the assumption
that L is in P must be false.

Given the cipher, we can construct a language that is in NP
but not in P. Thus, this particular form of cipher security -
where we can use an arbitrarily large key, encryption and
decryption are polytime in the key size and cryptanalysis is
exponential in the key size - requires that P != NP.

--Bryan

Subject: Re: Memo to the Amateur Cipher Designer
Date: 28 Oct 1998 08:43:07 GMT
From: olson@umbc.edu (Bryan G. Olson; CMSC (G))
Message-ID: <716ler$h90$1@news.umbc.edu>
References: <3634a751.2469260@news.io.com>
Newsgroups: sci.crypt
Lines: 42

Terry Ritter wrote:

: Bruce Schneier wrote:

: >Security is orthogonal to functionality. A cipher cannot deliver any
: >new advantages until it is considered strong. That's what makes this
: >discipline complicated.

: Apparently I have been unable to communicate the issue:

That may be, but Bruce understands the issues anyway.

[...]
: >No. The adversarial game of making and breaking is what makes
: >cryptography cryptography. I design; you break. You design; I break.
: >This is what cryptography is.

: I am not referring to legitimate thrust and parry of design and
: analysis, I am referring to exactly the sort of behavior in your (now
: deleted) anecdote. I claim:

: * The legitimate response to a design is a break.
: * The legitimate response to a fixed design is a break.
: * The legitimate response to a fixed fixed design is a break.

Absolutely. Please, please, start responding that way. Enough
of all the posts that respond to someone else's design by
pointing out features of your own designs. There are very few on
this group who actually devote time and effort to looking into
other peoples suggestions.

: Life is tough for cipher analyzers. It must be frustrating when
: newbies simply do not (no doubt interpreted as "will not") get the
: point.

Yes it is. The major "point" is that a cipher designer must
_be_ a cipher analyzer. That's what good designers spend most
of their time doing.

--Bryan

Subject: Re: Memo to the Amateur Cipher Designer
Date: Wed, 21 Oct 1998 05:22:08 GMT
From: dianelos@tecapro.com
Message-ID: <70jr20$k0l$1@nnrp1.dejanews.com>
References: <36292906.1151332@news.visi.com>
Newsgroups: sci.crypt
Lines: 99

In article <36292906.1151332@news.visi.com>,
  schneier@counterpane.com (Bruce Schneier) wrote:
>[...]
> "The best cryptographers around" break a lot of ciphers. The academic
> literature is littered with the carcasses of ciphers broken by their
> analyses.

I would rather not use the word "break" to describe the successful cryptanalysis of a cipher. If somebody found an attack against 3DES with 2^60 chosen plaintexts it would certainly be a great piece of cryptanalytic work but it would *not* mean that bank wire transfers could be BROKEN in any reasonable sense of the word. Again: if somebody found a way to compute the 3DES key with two known plaintexts and 15 seconds on a PC, how would we describe this attack - a "demolition" of 3DES or what? I think it would be better to say that a successful cryptanalysis discovered a weakness in a cipher. Any weakness can or should be mortal for a cipher, particularly if discovered with little effort or if that cipher can show no other advantages. Even so, the word "break" we should reserve for what the British did to the Enigma machine.

I know this is only semantics but still I think it is important. Security issues are slowly seeping into public awareness and it would be best not to use common words in a way that is contrary to their normal meaning.

> [...] Algorithms
> posted to Internet newsgroups by unknowns won't get a second glance.

My personal opinion is that in the future Internet newsgroups will be the most important medium for communicating ideas while peer reviewed publications, as we know them today, will be less and less important. This is not an either-or proposition. Stuffy, paper-based publications and chaotic, unstructured newsgroups will gravitate towards a medium that combines the best of the two worlds. Newsgroups do have enormous advantages: immediate and free movement of ideas is one of them. The other, I think, is that it is much cheaper for an author to be proven wrong in a newsgroup post; therefore people feel more free to publish crazy, less well thought out ideas. One of the seminal books in my life was Minsky's "Society of Mind". I think newsgroups will evolve to what is functionally a bigger mind. When I started toying with the idea of participating in the AES competition, I intended to post my basic idea and invite the sci.crypt crowd to participate as a group in the competition. When I finally started working on the submission it was too late. Pity - it would have been an

interesting experiment of the BIG MIND paradigm.

> [...] The
> cipher's strength is not in its design; anyone could design something
> like that. The strength is in its analysis.

Clearly there is no known formula or experiment that measures a cipher's strength. This leaves human based analysis as the only way to validate a cipher's strength today. Still, I see a problem here: Suppose cipher A is analyzed by good cryptographers and many interesting results are published even though no weakness is found. Cipher B is analyzed even more intensely by the same people, no weakness is found but neither is there anything interesting to report. B should be considered stronger than A, but in the current state of affairs the opposite would happen. My point is that published results do not necessarily indicate the quantity or quality of analysis done on a cipher.

Successful analysis depends not only on the cryptographer's previous experience or effort invested, but also on uncontrollable, unquantifiable factors such as inspiration or even luck. There is a real possibility that somebody will have an ingenious idea tomorrow that will demolish many ciphers we consider secure today. This is a terribly unfortunate situation: we are betting a significant part of tomorrows stability not so much on technology that is not *proven* but rather on technology we have no way to *test*. Meanwhile, NSA is not allowed to talk, and this, I think, is not wise.

>[...]
> 1. Describe your cipher using standard notation. This doesn't mean C
> code. There is established terminology in the literature. Learn it
> and use it; no one will learn your specialized terminology.

I don't completely agree. A standard representation often restricts the group of ideas that can comfortably be expressed. For example, FROG cannot be represented with the traditional hardware schematic; some algorithms can be represented more elegantly with unstructured pseudocode filled with GOTOs. A cipher is an algorithm - traditionally algorithms are described with pseudo-code or even with documented code written in Pascal, Lisp, C or some other well known language. I don't quite see why using C to describe a cipher is a bad idea. Anyway, I am splitting hairs here. It is self-evident that a cipher should be described in a clear way and that no specialized terminology should be used when none is needed.

--
http://www.tecapro.com
email: dianelos@tecapro.com

----------== Posted via Deja News, The Discussion Network ==----------
http://www.dejanews.com/ Search, Read, Discuss, or Start Your Own

---

Subject: Re: Memo to the Amateur Cipher Designer
Date: Thu, 22 Oct 1998 15:20:06 GMT
From: jsavard@tenMAPSONeerf.edmonton.ab.ca (John Savard)
Message-ID: <362f4c6d.832631@news.prosurfr.com>
References: <70jr20$k0l$1@nnrp1.dejanews.com>
Newsgroups: sci.crypt
Lines: 15

dianelos@tecapro.com wrote, in part:

> I would rather not use the word "break" to describe the successful
> cryptanalysis of a cipher. If somebody found an attack against
> 3DES with 2^60 chosen plaintexts it would certainly be a great
> piece of cryptanalytic work but it would *not* mean that bank wire
> transfers could be BROKEN in any reasonable sense of the word.

Well, that's a valid enough comment on terminology. However, with specific reference to the AES process, a cryptanalytic result that indicates a proposed cipher is less than _perfect_ is, quite properly, considered significant.

John Savard
http://members.xoom.com/quadibloc/index.html

Subject: Re: Memo to the Amateur Cipher Designer
Date: Fri, 23 Oct 1998 10:53:08 -0600
From: jgfunj@EnqvbSerrGrknf.pbz (W T Shaw)
Message-ID: <jgfunj-2310981053080001@dialup133.itexas.net>
References: <36305059.CAE8032F@stud.uni-muenchen.de>
  <jgfunj-2210981318390001@207.22.198.187>
Newsgroups: sci.crypt
Lines: 43

In article <36305059.CAE8032F@stud.uni-muenchen.de>, Mok-Kong Shen
<mok-kong.shen@stud.uni-muenchen.de> wrote:

> W T Shaw wrote:
> >

> > In article <362F34E7.118E41AC@stud.uni-muenchen.de>, Mok-Kong Shen
> > <mok-kong.shen@stud.uni-muenchen.de> wrote:
> > >
> > > dianelos@tecapro.com wrote:
> > > > For example, FROG cannot be represented with the traditional
> > > > hardware schematic; some algorithms can be represented more
> > > > elegantly with unstructured pseudocode filled with GOTOs. A cipher
> > > > is an algorithm - traditionally algorithms are described with
> > > > pseudo-code or even with documented code written in Pascal, Lisp,
> > > > C or some other well known language. I don't quite see why using C
> > > > to describe a cipher is a bad idea.
> > >
> > I get his idea, that to predicate a description or demonstration to an
> > artificially restrictive set of circumstances might preclude the simplest
> > or the most most meaningful one. Each media has its own built in
> > prejudices which might make things look harder than they are. I challenge
> > you to build a DES encryption machine with no electronics in it.
>
> Sorry that I don't yet understand. I thought what Dianelos wrote
> amounts to the following: FROG cannot be described with a program
> written in any of the currently used programming languages.

See above; clearly he said the describing a cipher in C would be OK with
him, but not in a traditional *hardware* schematic.

> But
> with what should FROG be properly described? Does one need a
> real-time programming language? There are hardware design languages,
> VHDL. Should FROG be described using these? I think we should
> await the answer from the designer of FROG rather than making
> speculations ourselves.
>
I expect him to agree with what he has said already as he is consistent.
--

Security by obscurity is a good description of bureaucratic spending.

Decrypt with ROT13 to get correct email address.

Subject: Re: Memo to the Amateur Cipher Designer
Date: Sat, 24 Oct 1998 03:31:06 GMT
From: dianelos@tecapro.com
Message-ID: <70rhlqock1@nnrp1.dejanews.com>
References: <36305059.CAE8032F@stud.uni-muenchen.de>
  <jgfunj-2210981318390001@207.22.198.187>
Newsgroups: sci.crypt
Lines: 65

In article <36305059.CAE8032F@stud.uni-muenchen.de>,
  Mok-Kong Shen <mok-kong.shen@stud.uni-muenchen.de> wrote:
> W T Shaw wrote:
> >
> > In article <362F34E7.118E41AC@stud.uni-muenchen.de>, Mok-Kong Shen
> > <mok-kong.shen@stud.uni-muenchen.de> wrote:
> > >
> > > dianelos@tecapro.com wrote:
> > > > For example, FROG cannot be represented with the traditional
> > > > hardware schematic; some algorithms can be represented more
> > > > elegantly with unstructured pseudocode filled with GOTOs. A cipher
> > > > is an algorithm - traditionally algorithms are described with
> > > > pseudo-code or even with documented code written in Pascal, Lisp,
> > > > C or some other well known language. I don't quite see why using C
> > > > to describe a cipher is a bad idea. Anyway, I am splitting hairs
> > >
> > > I don't quite catch your point. Does the sentence 'FROG cannot be ...'
> > > imply that it can't be described fully in C etc.?
> > >
> > I get his idea, that to predicate a description or demonstration to an
> > artificially restrictive set of circumstances might preclude the simplest
> > or the most most meaningful one. Each media has its own built in
> > prejudices which might make things look harder than they are. I challenge
> > you to build a DES encryption machine with no electronics in it.
>
> Sorry that I don't yet understand. I thought what Dianelos wrote
> amounts to the following: FROG cannot be described with a program
> written in any of the currently used programming languages. But
> with what should FROG be properly described? Does one need a
> real-time programming language? There are hardware design languages,
> VHDL. Should FROG be described using these? I think we should
> await the answer from the designer of FROG rather than making
> speculations ourselves.

In my original post I mentioned two examples of cases where the
traditional representation of an idea turns out not to be simplest.
Mok-kong thought the two examples are related - they are not.
Sorry for the ambiguity.

FROG cannot easily be represented by a hardware diagram because it
uses key dependent addresses. Hardware diagrams have fixed data paths.
In principle, of course, you can represent *any* algorithm either in
C or as hardware diagram. Sometimes a hardware diagram is the better
option. For example, it is easier to express a permutation using a
hardware diagram rather than in C. In general it is easier to express

concurrency with diagrams.

What representation you choose is not a trivial matter. If a cipher
designer always works sketching diagrams, in praxis he will
artificially limit the range of ideas that he will consider.
Also, changing back and forth from one representation to another
can be very useful sometimes. I recall how in school exams
I had to solve geometry problems using only Euclidean reasoning.
Well, I found out that I could often translate the problem into
vector algebra, easily solve it in this representation, and then
translate my proof, step by step, back into pure geometry.

--
http://www.tecapro.com
email: dianelos@tecapro.com

----------== Posted via Deja News, The Discussion Network ==----------
http://www.dejanews.com/ Search, Read, Discuss, or Start Your Own

---

Subject: Re: Memo to the Amateur Cipher Designer
Date: Sun, 25 Oct 1998 10:39:00 -0600
From: jgfunj@EnqvbSerrGrknf.pbz (W T Shaw)
Message-ID: <jgfunj-2510981039000001@dialup126.itexas.net>
References: <70rhlqock1@nnrp1.dejanews.com>
Newsgroups: sci.crypt
Lines: 21

In article <70rhlqock1@nnrp1.dejanews.com>, dianelos@tecapro.com wrote:
>
> What representation you choose is not a trivial matter. If a cipher
> designer always works sketching diagrams, in praxis he will
> artificially limit the range of ideas that he will consider.
> Also, changing back and forth from one representation to another
> can be very useful sometimes. I recall how in school exams
> I had to solve geometry problems using only Euclidean reasoning.
> Well, I found out that I could often translate the problem into
> vector algebra, easily solve it in this representation, and then
> translate my proof, step by step, back into pure geometry.
>
Hook or crook means anything that works is open for use. Having to work
things out in solely by careful appearing and impressive sounding logic
that may not be applicable to the real world is the essence of the
scientific Greek Tragedy.
--

Please excuse if there are multiple postings for my responses...I have no idea where
they come from as I only receive one confirmation for each posting from my
newsserver.

Decrypt with ROT13 to get correct email address.

Subject: Re: Memo to the Amateur Cipher Designer
Date: Mon, 26 Oct 1998 03:56:39 GMT
From: schneier@counterpane.com (Bruce Schneier)
Message-ID: <3633f23c.2025257@news.visi.com>
References: <70jr20$k0l$1@nnrp1.dejanews.com>
Newsgroups: sci.crypt
Lines: 61

On Wed, 21 Oct 1998 05:22:08 GMT, dianelos@tecapro.com wrote:
> I would rather not use the word "break" to describe the successful
> cryptanalysis of a cipher. If somebody found an attack against
> 3DES with 2^60 chosen plaintexts it would certainly be a great
> piece of cryptanalytic work but it would *not* mean that bank wire
> transfers could be BROKEN in any reasonable sense of the word.
> Again: if somebody found a way to compute the 3DES key with two
> known plaintexts and 15 seconds on a PC, how would we describe
> this attack - a "demolition" of 3DES or what? I think it would be
> better to say that a successful cryptanalysis discovered a
> weakness in a cipher. Any weakness can or should be mortal for a
> cipher, particularly if discovered with little effort or if that
> cipher can show no other advantages. Even so, the word "break" we
> should reserve for what the British did to the Enigma machine.

I agree that "break" is overused. No one will argue that most of the
breaks in the literature are what some of us call "academic breaks":
attacks that show theoretical weakness but cannot be used in real life
to break operational traffic. Prudence, of course, teaches that if
you have to choose between two ciphers, one with an academic break and
one without, you choose the one without.

>> [...] Algorithms
>> posted to Internet newsgroups by unknowns won't get a second glance.
>
> My personal opinion is that in the future Internet newsgroups will
> be the most important medium for communicating ideas while peer
> reviewed publications, as we know them today, will be less and
> less important.

Not a chance. In a world where everyone is a publisher, editors
become even more important.

>> 1. Describe your cipher using standard notation. This doesn't mean C
>> code. There is established terminology in the literature. Learn it
>> and use it; no one will learn your specialized terminology.
>
>   I don't completely agree. A standard representation often
>   restricts the group of ideas that can comfortably be expressed.
>   For example, FROG cannot be represented with the traditional
>   hardware schematic; some algorithms can be represented more
>   elegantly with unstructured pseudocode filled with GOTOs. A cipher
>   is an algorithm - traditionally algorithms are described with
>   pseudo-code or even with documented code written in Pascal, Lisp,
>   C or some other well known language. I don't quite see why using C
>   to describe a cipher is a bad idea. Anyway, I am splitting hairs
>   here. It is self-evident that a cipher should be described in a
>   clear way and that no specialized terminology should be used when
>   none is needed.

I disagree with your disagreement, but I expect it is a semantic
distinction. "The established terminology in the literature" is not C
code, assembly code, a hardware schematic, or any implementation
language. The established terminology for cryptography is mathematics.
I can describe FROG mathematically, and so can you.

Bruce
****
Bruce Schneier, President, Counterpane Systems   Phone: 612-823-1098
101 E Minnehaha Parkway, Minneapolis, MN 55419   Fax: 612-823-1590
           Free crypto newsletter. See: http://www.counterpane.com

---

Subject: Re: Memo to the Amateur Cipher Designer
Date: Mon, 26 Oct 1998 14:21:00 +0100
From: Mok-Kong Shen <mok-kong.shen@stud.uni-muenchen.de>
Message-ID: <3634773C.B78AB011@stud.uni-muenchen.de>
References: <3633f23c.2025257@news.visi.com>
Newsgroups: sci.crypt
Lines: 21

Bruce Schneier wrote:
>
> On Wed, 21 Oct 1998 05:22:08 GMT, dianelos@tecapro.com wrote:

> >   here. It is self-evident that a cipher should be described in a
> >   clear way and that no specialized terminology should be used when
> >   none is needed.
>
> I disagree with your disagreement, but I expect it is a semantic
> distinction. "The established terminology in the literature" is not C
> code, assembly code, a hardware schematic, or any implementation
> language. The established terminology for cryptography is mathematics.
> I can describe FROG mathematically, and so can you.

I think that the economy of description decides to some extent
which way of presentation is to be prefered. As far as I know, in
mathematics one rarely (almost never) writes proofs in terms of
formal logic calculus, because that, although more rigorous, is much
more tedious.

M. K. Shen

Subject: Re: Memo to the Amateur Cipher Designer
Date: Mon, 26 Oct 1998 16:46:24 GMT
From: ritter@io.com (Terry Ritter)
Message-ID: <3634a75a.2478727@news.io.com>
References: <3633f23c.2025257@news.visi.com>
Newsgroups: sci.crypt
Lines: 72

On Mon, 26 Oct 1998 03:56:39 GMT, in <3633f23c.2025257@news.visi.com>,
in sci.crypt schneier@counterpane.com (Bruce Schneier) wrote:

>[...]
>>> [...] Algorithms
>>> posted to Internet newsgroups by unknowns won't get a second glance.
>>
>>   My personal opinion is that in the future Internet newsgroups will
>>   be the most important medium for communicating ideas while peer
>>   reviewed publications, as we know them today, will be less and
>>   less important.
>
>Not a chance. In a world where everyone is a publisher, editors
>become even more important.

I recently posted a quote about this from the current IEEE Spectrum in
another thread. Basically the idea is that the world is moving *away*
from intermediaries who filter and decide for us, to the end-user (of
clothes, of technical articles, etc.) surveying it all, and making the
decision on what to select. One can argue how far this will go, but
the time is past when somebody could just sit and wait for the
articles to arrive and thus be assured of knowing the field.

```
>>> 1.  Describe your cipher using standard notation.  This doesn't mean C
>>> code.  There is established terminology in the literature.  Learn it
>>> and use it; no one will learn your specialized terminology.
>>
>>    I don't completely agree. A standard representation often
>>    restricts the group of ideas that can comfortably be expressed.
>>    For example, FROG cannot be represented with the traditional
>>    hardware schematic; some algorithms can be represented more
>>    elegantly with unstructured pseudocode filled with GOTOs. A cipher
>>    is an algorithm - traditionally algorithms are described with
>>    pseudo-code or even with documented code written in Pascal, Lisp,
>>    C or some other well known language. I don't quite see why using C
>>    to describe a cipher is a bad idea. Anyway, I am splitting hairs
>>    here. It is self-evident that a cipher should be described in a
>>    clear way and that no specialized terminology should be used when
>>    none is needed.
>
>I disagree with your disagreement, but I expect it is a semantic
>distinction.  "The established terminology in the literature" is not C
>code, assembly code, a hardware schematic, or any implementation
>language. The established terminology for cryptography is mathematics.
>I can describe FROG mathematically, and so can you.

Sure, we can describe *any* logic machine mathematically, but why
would one want to?  If math is a great advantage in understanding
logic machines, why are logic machines not generally described that
way?  Why?  Because schematics can be clearer, that's why.  And
clarity in the presentation is exactly what we want.

Now, there are ciphers for which math is the appropriate description:
Number-theoretic ciphers and so on.  Math is at the heart of those
ciphers, and governs how they work.  To understand them, math is the
appropriate notation.

Most symmetric designs, however, are not number-theoretic, nor do they
have any coherent mathematical theory.  Yes, one could cast them into
math, but why?  Without the underlying mathematical theory, where is
the advantage?  Indeed, translating the arbitrary machine from its
design notation into another notation seems likely to hide the very
issues on which the cipher is based, and those are exactly the issues
which might be most important for analysis.

---
Terry Ritter   ritter@io.com   http://www.io.com/~ritter/
Crypto Glossary   http://www.io.com/~ritter/GLOSSARY.HTM
```

```
Subject: Re: Memo to the Amateur Cipher Designer
Date: Wed, 28 Oct 1998 10:56:20 +0000
From: Frank O'Dwyer &LT;fod@brd.ie>
Message-ID: &LT;3636F854.65C31548@brd.ie>
References: &LT;3633f23c.2025257@news.visi.com>
Newsgroups: sci.crypt
Lines: 16

Bruce Schneier wrote:
> On Wed, 21 Oct 1998 05:22:08 GMT, dianelos@tecapro.com wrote:
> >    My personal opinion is that in the future Internet newsgroups will
> >    be the most important medium for communicating ideas while peer
> >    reviewed publications, as we know them today, will be less and
> >    less important.
>
> Not a chance.  In a world where everyone is a publisher, editors
> become even more important.

I think the key phrase above is "peer reviewed publications, as we know
them today". In a world where everyone can be a publisher, everyone can
be an editor too.

Cheers,
Frank O'Dwyer.
```

```
Subject: Re: Memo to the Amateur Cipher Designer
Date: 28 Oct 1998 09:41:52 -0500
From: juola@mathcs.duq.edu (Patrick Juola)
Message-ID: &LT;717afg$jir$1@quine.mathcs.duq.edu>
References: &LT;3636F854.65C31548@brd.ie>
Newsgroups: sci.crypt
Lines: 25

In article &LT;3636F854.65C31548@brd.ie>, Frank O'Dwyer  &LT;fod@brd.ie> wrote:
>Bruce Schneier wrote:
>> On Wed, 21 Oct 1998 05:22:08 GMT, dianelos@tecapro.com wrote:
>> >    My personal opinion is that in the future Internet newsgroups will
>> >    be the most important medium for communicating ideas while peer
>> >    reviewed publications, as we know them today, will be less and
>> >    less important.
>>
>> Not a chance.  In a world where everyone is a publisher, editors
>> become even more important.
>
```

>I think the key phrase above is "peer reviewed publications, as we know
>them today". In a world where everyone can be a publisher, everyone can
>be an editor too.

Which implies that the value of good, worthwhile editing will continue
to climb, just as the value of good *writing* has been climbing since
the development of the Internet.

I suspect that peer-reviewed publications have become more and more
important over the last 20 years as information channels, as the
informal channels (e.g. Karp, p.c.) have gotten more and more clogged
by garbage.

-kitten

Subject: Re: Memo to the Amateur Cipher Designer
Date: Wed, 28 Oct 1998 18:32:40 GMT
From: aquiranx@goliat.ugr.es (Gurripato (x=nospam))
Message-ID: <36375758.26994734@news.cica.es>
References: <3633f23c.2025257@news.visi.com>
Newsgroups: sci.crypt
Lines: 13

On Mon, 26 Oct 1998 03:56:39 GMT, schneier@counterpane.com (Bruce Schneier)
wrote:

>I agree that "break" is overused. No one will argue that most of the
>breaks in the literature are what some of us call "academic breaks":
>attacks that show theoretical weakness but cannot be used in real life
>to break operational traffic. Prudence, of course, teaches that if
>you have to choose between two ciphers, one with an academic break and
>one without, you choose the one without.
>
How would you then best describe Dobbertin´s attack on the
compression function of MD5? Does it go all the way to demolition, plan
brack, or just academic break?

Subject: Re: Memo to the Amateur Cipher Designer
Date: Tue, 27 Oct 1998 11:16:48 GMT
From: sjmz@hplb.hpl.hp.com (Stefek Zaba)
Message-ID: <F1HGo1.GIC@hplb.hpl.hp.com>
References: <jgfunj-2210981318390001@207.22.198.187>
<362F34E7.118E41AC@stud.uni-muenchen.de>
<70jr20$k0l$1@nnrp1.dejanews.com>
Newsgroups: sci.crypt
Lines: 23

In sci.crypt, W T Shaw (jgfunj@EnqvbSerrGrknf.pbz) wrote:
>
> I get his idea, that to predicate a description or demonstration to an
> artificially restrictive set of circumstances might preclude the simplest
> or the most most meaningful one. Each media has its own built in
> prejudices which might make things look harder than they are. I challenge
> you to build a DES encryption machine with no electronics in it.
>
OK: consider a city full of Chinese people, rigorously following written
instructions on the handling of small (8-byte, to be concrete) amounts of
information in particular systematic ways. (Yes, a group of individuals
running round with an internal monologue which if dragged first from
mentalese to Cantonese and thus to English might be rendered as "I'm an
S-box! I'm an S-box!", "I'm an transposer! I'm a transposer!", and the like.)
They pass the outputs of their rule-following to designated, possibly
conditionally-different, individuals. Such a small city can compute DES
encryptions/decryptions without a semiconductor in sight.

And that estimable Mr Searle will explain why each of them must be shot
after performing their role more than a few times, as they now understand
not only block cipher design but the content of the encrypted message :-)

Cheerski, Stefek "I Was A Chinese Sex Slave" Z

Subject: Re: Memo to the Amateur Cipher Designer
Date: Thu, 22 Oct 1998 23:31:33 GMT
From: mr.i.o.yankle@anagrams.r.us (Mr. I. O. Yankle)
Message-ID: <362fbdbc.162209710@news.alt.net>
References: <36292906.1151332@news.visi.com>
Newsgroups: sci.crypt
Lines: 8

When I first read "Memo to the Amateur Cipher Designer" in Bruce Schneier's
CRYPTO-GRAM, it was so clearly true and sensible to me that I expected it
to gain immediate acceptance on sci.crypt and to even gain the status of
"required reading". I still hope that this will be the case, but I can see
now that it will take some time.
--
"Mr. I. O. Yankle" better known as 0279.654831@mail.serve.com.
01 2 3 456789 <- Use this key to decode my email address.

Subject: Re: Memo to the Amateur Cipher Designer

Date: Thu, 22 Oct 1998 21:41:08 -0600
From: jgfunj@EnqvbSerrGrknf.pbz (W T Shaw)
Message-ID: <jgfunj-2210982141230001@207.101.116.111>
References: <362fbdbc.162209710@news.alt.net>
Newsgroups: sci.crypt
Lines: 20

In article <362fbdbc.162209710@news.alt.net>, mr.i.o.yankle@anagrams.r.us
(Mr. I. O. Yankle) wrote:

> When I first read "Memo to the Amateur Cipher Designer" in Bruce Schneier's
> CRYPTO-GRAM, it was so clearly true and sensible to me that I expected it
> to gain immediate acceptance on sci.crypt and to even gain the status of
> "required reading". I still hope that this will be the case, but I can see
> now that it will take some time.

I could be that the what is so clearly true and sensible to you is not
necessarily so. Indeed, many of the thoughts have been expressed before.
It is rather that the devil is as always in the details, and the audience
here is not immune to nit picking at generalizations which are best
accepted by those who know little or nothing about the subject. Such
popularistic wisdom is best spent elsewhere.
--

Passing a budgit with obscure items is bad; preventing government payment for birth
control while authorizing millions for viagra lets us focus on the hard facts of
prevalent sexism.

Decrypt with ROT13 to get correct email address.

Subject: Re: Memo to the Amateur Cipher Designer
Date: Mon, 26 Oct 1998 16:45:49 GMT
From: ritter@io.com (Terry Ritter)
Message-ID: <3634a72a.2430860@news.io.com>
References: <362fbdbc.162209710@news.alt.net>
Newsgroups: sci.crypt
Lines: 35

On Thu, 22 Oct 1998 23:31:33 GMT, in
<362fbdbc.162209710@news.alt.net>, in sci.crypt
mr.i.o.yankle@anagrams.r.us (Mr. I. O. Yankle) wrote:

>When I first read "Memo to the Amateur Cipher Designer" in Bruce Schneier's
>CRYPTO-GRAM, it was so clearly true and sensible to me that I expected it
>to gain immediate acceptance on sci.crypt and to even gain the status of
>"required reading". I still hope that this will be the case, but I can see
>now that it will take some time.

I would hope that anyone reading Schneier's article would recognize
that it is seriously flawed in many ways. Here are some interesting
points from the article:

* Someone with a good idea and presentation will have trouble getting
published if they are not part of "the crypto clique."

* The way to handle those with less knowledge is to demonstrate how
much smarter we are so they will go away.

* Extensive cryptanalysis can prove cipher strength.

From a whole list of appalling ideas, this last is perhaps the most
breathtaking, as it goes to the fundamental basis of modern
cryptography by a renowned expert in the field.

Perhaps you should review my response of Tue, 20 Oct 1998 00:40:21 GMT
in message id 362bdbc6.3212829@news.io.com.

Terry Ritter ritter@io.com http://www.io.com/~ritter/
Crypto Glossary http://www.io.com/~ritter/GLOSSARY.HTM

Subject: Re: Memo to the Amateur Cipher Designer
Date: Tue, 27 Oct 1998 02:36:58 GMT
From: kery.minola@anagrams.r.us (Kery Minola)
Message-ID: <36352680.84964040@news.gate.net>
References: <3634a72a.2430860@news.io.com>
Newsgroups: sci.crypt
Lines: 58

ritter@io.com (Terry Ritter) wrote:

>I would hope that anyone reading Schneier's article would recognize
>that it is seriously flawed in many ways. Here are some interesting
>points from the article:

>* Someone with a good idea and presentation will have trouble getting
>published if they are not part of "the crypto clique."

Bruce Schneier's article does not mention a "clique", perhaps that's your

term for the scientific community. That you must show that you know what you're talking about before people will listen to you is a fact of life. He merely reported it.

>* The way to handle those with less knowledge is to demonstrate how
>much smarter we are so they will go away.

The way to handle those with less knowledge is to show them how much they have yet to learn and to point the way.

>* Extensive cryptanalysis can prove cipher strength.

I had no problem understanding what he meant. He meant that extensive cryptanalysis of a cipher is the best evidence of strength that you can hope for.

>From a whole list of appalling ideas, this last is perhaps the most
>breathtaking, as it goes to the fundamental basis of modern
>cryptography by a renowned expert in the field.

Here's what he said:

>What is hard is creating an algorithm that no one else can break, even
>after years of analysis. And the only way to prove that is to subject
>the algorithm to years of analysis by the best cryptographers around.

You are really grasping at straws if you are trying to pin him down to the literal, mathematical meaning of "prove". Are you suggesting that Bruce Schneier is totally oblivious to the weekly sci.crypt discussions about how the O.T.P. is the only provably secure cipher? Obviously he meant "prove" in the everyday sense of "beyond a reasonable doubt".

>Perhaps you should review my response of Tue, 20 Oct 1998 00:40:21 GMT
>in message id 362bdbc6.3212829@news.io.com.

Yes, I read that message as well.

My impression is that the amateur cryptologists here are currently in denial of what they know is true. In time, I believe the document "Memo to the Amateur Cipher Designer" will become a handy countermeasure to use against the annoying posts of gibberish that we see here, which are always accompanied by an arrogant challenge to break the code.

It's the truth. Face it and embrace it!
--
"Kery Minola" better known as 4501.693872@mail.serve.com.
0123 456789 <- Use this key to decode my email address.
5 X 5 Poker - http://www.serve.com/games/

Subject: Re: Memo to the Amateur Cipher Designer
Date: 27 Oct 1998 04:04:19 GMT
From: caj@baker.math.niu.edu (Xcott Craver)
Message-ID: <713go3ora1@gannett.math.niu.edu>
References: <3634a72a.2430860@news.io.com>
Newsgroups: sci.crypt
Lines: 26

Terry Ritter <ritter@io.com> wrote:
>
>I would hope that anyone reading Schneier's article would recognize
>that it is seriously flawed in many ways. Here are some interesting
>points from the article:
>
>* Someone with a good idea and presentation will have trouble getting
>published if they are not part of "the crypto clique."

Well, they say that every reading is a misreading.

Not only did the memo NOT say this, but it outlined how to get one's foot in the door via publishing attacks. I can verify that this strategy works like a charm. Publishing is not easy or perfect, but the accusation of an entrenched scientific clique is the stuff of UFO cover-up theories, creationism and flawed proofs of Fermat's last theorem.

>* Extensive cryptanalysis can prove cipher strength.

!!! It's obvious that the memo did not mean "prove" in the strict mathematical sense, but in the empirical sense. Cryptography being a science, I don't exactly see anything wrong with using the SCIENTIFIC METHOD.

-Caj

Subject: Re: Memo to the Amateur Cipher Designer
Date: Tue, 27 Oct 1998 05:59:54 GMT
From: "Douglas A. Gwyn" <DAGwyn@null.net>
Message-ID: <36356103.9BC3B1FA@null.net>
References: <713go3ora1@gannett.math.niu.edu>
Newsgroups: sci.crypt
Lines: 15

Xcott Craver wrote:
> !!! It's obvious that the memo did not mean "prove" in the
> strict mathematical sense, but in the empirical sense.

The trouble is, with cryptography the protected message seems
absolutely secure against eavesdropping, until a cryptanalytic
breakthrough occurs, after which it is horribly insecure.
The "empirical proof" means very little since it can't allow
for the eavesdropper's cryptanalytic abilities.

> Cryptography being a science, I don't exactly see anything
> wrong with using the SCIENTIFIC METHOD.

There is a huge difference between studying nature and
analyzing products of the mind of man.

Subject: Re: Memo to the Amateur Cipher Designer
Date: 27 Oct 1998 19:05:14 GMT
From: caj@baker.math.niu.edu (Xcott Craver)
Message-ID: <7155ha$nt3$1@gannett.math.niu.edu>
References: <36356103.9BC3B1FA@null.net>
Newsgroups: sci.crypt
Lines: 25

Douglas A. Gwyn <DAGwyn@null.net> wrote:
>Xcott Craver wrote:
>
>> Cryptography being a science, I don't exactly see anything
>> wrong with using the SCIENTIFIC METHOD.
>
>There is a huge difference between studying nature and
>analyzing products of the mind of man.

  Are you suggesting that we should use something other than the
  scientific method? Nobody claims that basing a conclusion
  on empirical evidence is perfect, or even safe; but what
  alternative?

  Further, whether or not mathematical constructs are
  "the products of the mind of man" has been debated, hotly,
  for as long as there have been philosophers. Mathematical
  realists would consider the study of ciphers literally the study
  of the universe around us --- just the intangible part of
  the universe.

  Finally, why on Earth should the scientific method be
  disqualified in the case of studying the products of the human
  mind? Do you know something that all the psychologists in the
  world don't?

Subject: Re: Memo to the Amateur Cipher Designer
Date: Tue, 27 Oct 1998 20:50:22 -0600
From: jgfunj@EnqvbSerrGrknf.pbz (W T Shaw)
Message-ID: <jgfunj-2710982050220001@dialup146.itexas.net>
References: <7155ha$nt3$1@gannett.math.niu.edu>
Newsgroups: sci.crypt
Lines: 42

In article <7155ha$nt3$1@gannett.math.niu.edu>, caj@baker.math.niu.edu
(Xcott Craver) wrote:

> Douglas A. Gwyn <DAGwyn@null.net> wrote:
> >Xcott Craver wrote:
> >
> >> Cryptography being a science, I don't exactly see anything
> >> wrong with using the SCIENTIFIC METHOD.
> >
> >There is a huge difference between studying nature and
> >analyzing products of the mind of man.
>
> Are you suggesting that we should use something other than the
> scientific method? Nobody claims that basing a conclusion
> on empirical evidence is perfect, or even safe; but what
> alternative?
>
Science uses lots of methods, including one actually called *the
scientific method*. To demand a single route to the truth is to prejudice
against truths that may not be so conform to that path. This is the
essence of what is wrong with what Bruce advocates, which is the same old
tired argument we have heard for ages.

Sophisocated groups, like individuals, can be entirely wrong. To use
popularity and acceptance as measures to oppose the introduction of new
ideas moves from scientific humility to politics, which is seldom a friend
to basic science. You must realize your own prejudices before you can be
worthy of judging the motives of others; you should be ready to accept
good data, even if it conflicts with that which you have perviously
accepted.

If anyone really is on a search for truth, they will not press artificial
hurdles in anyones way. This means that informal means should not be
ignored, because formalism by definition tends to be prejudicial and

self-serving, rationalizing the importance of its own existence. It even
cause well-meaning people to lose there handle on what science is all
about, if they ever considered finding truth as a lofty imperative.
--

Please excuse if there are multiple postings for my responses...I have no idea where
they come from as I only receive one confirmation for each posting from my
newsserver.

Decrypt with ROT13 to get correct email address.

Subject: Re: Memo to the Amateur Cipher Designer
Date: 28 Oct 98 05:50:37 GMT
From: rigoleto@table.jps.net (Mike Zorn)
Message-ID: <3636b0ad.0@blushng.jps.net>
References: <jgfunj-2710982050220001@dialup146.itexas.net>
Newsgroups: sci.crypt
Lines: 17

W T Shaw (jgfunj@EnqvbSerrGrknf.pbz) wrote:
: In article <7155ha$nt3$1@gannett.math.niu.edu>, caj@baker.math.niu.edu
: (Xcott Craver) wrote:
: > Douglas A. Gwyn <DAGwyn@null.net> wrote:
: > >Xcott Craver wrote:
: > >> Cryptography being a science, I don't exactly see anything
: > >> wrong with using the SCIENTIFIC METHOD.
: > >There is a huge difference between studying nature and
: > >analyzing products of the mind of man.
: > Are you suggesting that we should use something other than the
: > scientific method? Nobody claims that basing a conclusion
: > on empirical evidence is perfect, or even safe; but what
: > alternative?
As an example, the benzene ring was not discovered by the 'scientific
method'. (on the other hand, we can't all be Kekule.) The SM is a powerful
tool, and it is quite useful - it's just not the olny one.
Mike Zorn

Subject: Re: Memo to the Amateur Cipher Designer
Date: Wed, 28 Oct 1998 17:09:04 GMT
From: stefekz@netcom.com (Stefek Zaba)
Message-ID: <stefekzF1Jrn4.HGK@netcom.com>
References: <3636b0ad.0@blushng.jps.net>
Newsgroups: sci.crypt
Lines: 23

Mike Zorn (rigoleto@table.jps.net) wrote:

: As an example, the benzene ring was not discovered by the 'scientific
: method'. (on the other hand, we can't all be Kekule.) The SM is a powerful
: tool, and it is quite useful - it's just not the olny one.

Kekule's *intuition* about a possible structure for benzene may be implausible
to explain as a deductive process: however, the observational data which K
was trying to explain, and subsequent observations on the behaviour of
benzene, *are* applications of "the scientifdic method". Were Kekule doing
abstract drawing, he could doodle a hexagon with thickened vertices, and
leave appreciation to the aesthetic sense of his intended audience: but as
a falsifiable hypothesis about the structure of a benzene molecule, such a
sketch must also agree with observed data.

Similarly, you, I, or my cat can come up with a block cipher design, and
can call it "elegant", "minimal", "beautiful", "secure", or "toffee-flavoured".
As an act of private creation it's an interesting artefact. But if we want
it to be used, either in practice or as a case study of potential design
principles, we must expect that design to be subjected to inspection, testing,
analysis, attack, and other procedures which play the role of "observations".

Cheers, Stefek

Subject: Re: Memo to the Amateur Cipher Designer
Date: Wed, 28 Oct 1998 19:12:39 GMT
From: ritter@io.com (Terry Ritter)
Message-ID: <36376c92.5645084@news.io.com>
References: <stefekzF1Jrn4.HGK@netcom.com>
Newsgroups: sci.crypt
Lines: 34

On Wed, 28 Oct 1998 17:09:04 GMT, in <stefekzF1Jrn4.HGK@netcom.com>,
in sci.crypt stefekz@netcom.com (Stefek Zaba) wrote:

>[...]
>Similarly, you, I, or my cat can come up with a block cipher design, and
>can call it "elegant", "minimal", "beautiful", "secure", or "toffee-flavoured".
>As an act of private creation it's an interesting artefact. But if we want
>it to be used, either in practice or as a case study of potential design
>principles, we must expect that design to be subjected to inspection, testing,
>analysis, attack, and other procedures which play the role of "observations".

Note, however, that cryptographic "observations" do not have the same

flavor as the usual scientific investigation: The thing we wish to show -- strength -- cannot be shown by observation, and also cannot be proven as a result of observations.

In normal science we innovate experiments to prove a result and get a new fact. In cryptography, we innovate experiments to prove a failure, and with a lack of failure we somehow leap to a conclusion of strength. This is a faulty leap. Crucially, the inability to break a cipher after much effort says nothing about its "real" strength.

Indeed, the conclusion of strength after analysis is *so* faulty that an old, well-accepted cipher could in fact be weaker than a new cipher with an obvious break. Even a broken new cipher could in fact be the better choice. Not being able to know strength is *really* not being able to know, and that is what we have.

Terry Ritter ritter@io.com http://www.io.com/~ritter/
Crypto Glossary http://www.io.com/~ritter/GLOSSARY.HTM

Subject: Re: Memo to the Amateur Cipher Designer
Date: Wed, 28 Oct 1998 23:48:40 GMT
From: dscott@networkusa.net
Message-ID: <718ago$r61$1@nnrp1.dejanews.com>
References: <stefekzF1Jrn4.HGK@netcom.com>
Newsgroups: sci.crypt
Lines: 35

In article <stefekzF1Jrn4.HGK@netcom.com>,
stefekz@netcom.com (Stefek Zaba) wrote:
> Mike Zorn (rigoleto@table.jps.net) wrote:
> ...

> Similarly, you, I, or my cat can come up with a block cipher design, and
> can call it "elegant", "minimal", "beautiful", "secure", or
"toffee-flavoured".
> As an act of private creation it's an interesting artefact. But if we want
> it to be used, either in practice or as a case study of potential design
> principles, we must expect that design to be subjected to inspection, testing,
> analysis, attack, and other procedures which play the role of "observations".
>
> Cheers, Stefek
>

ACtually if you come up with a good cipher you will not get it tested since they try to keep the rank of phony experts quite small. They may toss a bone from there high perch be decypting some easy old stuff but if it is good they will steal it modify it a little and try to take the credit for them selves. When did the experts start talking about all or nothing crypto not to long ago was it?
I have need told BS Bruce hates my guts. He can joke about my code but I am a thron in his side. He would show that it is easy to break mine if he could. The facts are he and his clan can't. But they may post some babble one of these days to confuse the masses.

--
http://cryptography.org/cgi-bin/crypto.cgi/Misc/scott19u.zip
ftp search for scott*u.zip in norway

----------== Posted via Deja News, The Discussion Network ==----------
http://www.dejanews.com/ Search, Read, Discuss, or Start Your Own

---

Subject: Re: Memo to the Amateur Cipher Designer
Date: Wed, 28 Oct 1998 05:43:00 GMT
From: "Douglas A. Gwyn" <DAGwyn@null.net>
Message-ID: <3636AE8A.4BC508A4@null.net>
References: <7155ha$nt3$1@gannett.math.niu.edu>
Newsgroups: sci.crypt
Lines: 37

Xcott Craver wrote:
> Douglas A. Gwyn <DAGwyn@null.net> wrote:
> >Xcott Craver wrote:
> >
> >> Cryptography being a science, I don't exactly see anything
> >> wrong with using the SCIENTIFIC METHOD.
> >There is a huge difference between studying nature and
> >analyzing products of the mind of man.
> Are you suggesting that we should use something other than the
> scientific method?

Sure. Merriam-Webster's Collegiate Dictionary says:
scientific method n (1854): principles and procedures for the
systematic pursuit of knowledge involving the recognition and
formulation of a problem, the collection of data through
observation and experiment, and the formulation and testing
of hypotheses.
This clearly is an empirical method (observation and experiment)

and so is less relevant to mathematical disciplines than it is to
physical science.

In other words, the so-called "scientific method" is but one tool
in our epistemological arsenal and ought not to be applied where
it is ineffective.

> Further, whether or not mathematical constructs are
> "the products of the mind of man" has been debated, hotly,
> for as long as there have been philosophers.

Yeah, but the Platonists are wrong.

> Finally, why on Earth should the scientific method be
> disqualified in the case of studying the products of the human
> mind? Do you know something that all the psychologists in the
> world don't?

That would be no great trick.

Subject: Re: Memo to the Amateur Cipher Designer
Date: 28 Oct 1998 17:15:07 GMT
From: caj@baker.math.niu.edu (Xcott Craver)
Message-ID: <717jer$3c2$1@gannett.math.niu.edu>
References: <3636AE8A.4BC508A4@null.net>
Newsgroups: sci.crypt
Lines: 36

Douglas A. Gwyn <DAGwyn@null.net> wrote:

>Xcott Craver wrote:
>>
>> Are you suggesting that we should use something other than the
>> scientific method?
>
>Sure. Merriam-Webster's Collegiate Dictionary says:
[...]
>This clearly is an empirical method (observation and experiment)
>and so is less relevant to mathematical disciplines than it is to
>physical science.

Well, so what do you suggest as an alternative?

Remember, this was about how one decides to trust a cipher
as "secure." The empirical method is to pick one most resistant
to analysis. Your suggestion?

>> Further, whether or not mathematical constructs are
>> "the products of the mind of man" has been debated, hotly,
>> for as long as there have been philosophers.
>
>Yeah, but the Platonists are wrong.

Care to explain why, and put half the philosophy faculty in the
world out of business?

'Sides, I'm talking about mathematical Realism. Slightly different
from Platonism, and a LARGE number of mathematicians are realists.
Surely you're a smart guy, especially if you know a better way
to judge a cipher's security other than empirically, but you're
implicitly declaring yourself smarter than a large number of people.
Call me an empiricist, but I'd like to see some data.

-Caj

Subject: Re: Memo to the Amateur Cipher Designer
Date: 28 Oct 1998 08:23:05 GMT
From: olson@umbc.edu (Bryan G. Olson; CMSC (G))
Message-ID: <716k99gqm1@news.umbc.edu>
References: <3634a72a.2430860@news.io.com>
Newsgroups: sci.crypt
Lines: 67

Terry Ritter wrote:

: Mr. I. O. Yankle wrote:

: >When I first read "Memo to the Amateur Cipher Designer" in Bruce Schneier's
: >CRYPTO-GRAM, it was so clearly true and sensible to me that I expected it
: >to gain immediate acceptance on sci.crypt and to even gain the status of
: >"required reading".

Absolutely. I agreed with Mr. Ritter on one point, but
clearly Bruce got at least a 95%.

: I would hope that anyone reading Schneier's article would recognize
: that it is seriously flawed in many ways. Here are some interesting
: points from the article:

: * Someone with a good idea and presentation will have trouble getting
: published if they are not part of "the crypto clique."

That's not really what he said. He recommended beginning with cryptanalysis which is more likely to be publishable than designs. Note that he said most conferences and workshops won't accept design from unknowns "without extensive analysis". The only unfairness is the suggestion that the same forums present designs from established experts without extensive analysis.

: * The way to handle those with less knowledge is to demonstrate how
: much smarter we are so they will go away.

That's not in Bruce's paper.

: * Extensive cryptanalysis can prove cipher strength.

Again, what you're saying isn't what Bruce said, and I think you know it.

Bruce wrote:

> What is hard is creating an algorithm that no one else can break, even after years of analysis. And the only way to prove that is to subject the algorithm to years of analysis by the best cryptographers around.

No one claimed that failure to break a cipher results in some kind of mathematical theorem saying it's strong. What Bruce did say is the _only_ way we can know a cipher stands up to years of cryptanalysis by actually subjecting it to years of cryptanalysis.

: From a whole list of appalling ideas, this last is perhaps the most
: breathtaking, as it goes to the fundamental basis of modern
: cryptography by a renowned expert in the field.

You can misinterpret it or whine about it all you want, but what Bruce actually wrote is true.

: Perhaps you should review my response of Tue, 20 Oct 1998 00:40:21 GMT
: in message id 362bdbc6.3212829@news.io.com.

Or perhaps you should go back and read the memo. There are clues in it for you.

--Bryan

Subject: Re: Memo to the Amateur Cipher Designer
Date: 26 Oct 1998 10:57:43 -0500
From: juola@mathcs.duq.edu (Patrick Juola)
Message-ID: <71265nsbd1@quine.mathcs.duq.edu>
References: <jgfunj-2610980949060001@dialup165.itexas.net>
   <711sa1$f74$5@korai.cygnus.co.uk>
Newsgroups: sci.crypt
Lines: 28

In article <jgfunj-2610980949060001@dialup165.itexas.net>,
W T Shaw <jgfunj@EnqvbSerrGrknf.pbz> wrote:
>In article <711sa1$f74$5@korai.cygnus.co.uk>, aph@cygnus.remove.co.uk
>(Andrew Haley) wrote:
>>
>> I don't see the relevance of this. The best evidence of a
>> one-year-old's thinking is the way in which the communicate.
>>
>The question of whether language is necessary for complex thought is one
>of ongoing debate and research; it is not simply answered. Some would
>jump to the conclusion that problem solving could not exist in isolation.
>I've been around too many animals that learned, even wild ones where
>instinct could not be blamed for resulting elaborate behavior.

It's also completely irrelevant to the discussion at hand, unless one is suggesting that one's goldfish is the designer of a cryptographic algorithm. Whether ``language'' and ``complex thought'' are separable in the abstract is one question -- but in practical terms, every human is capable of both and usually does both at the same time. It's quite reasonable to use a person's ability to write clearly as a gauge for his/her ability to *think* clearly, given the observed high correlation between the two.

The actual statistics of correlation are left as an exercise to be pulled out of any Psychology 101 textbook. They're out there, believe me.

-kitten

Subject: Re: Memo to the Amateur Cipher Designer
Date: Tue, 27 Oct 1998 16:56:18 +0100
From: Mok-Kong Shen <mok-kong.shen@stud.uni-muenchen.de>
Message-ID: <3635ED22.2F970FC3@stud.uni-muenchen.de>
References: <36292906.1151332@news.visi.com>

Newsgroups: sci.crypt
Lines: 113

Bruce Schneier wrote:
>
> This was in the October CRYPTO-GRAM, but I thought I'd run it through
> sci.crypt, since so many people seem to be asking questions on the
> topic.

The present thread is the biggest one I have ever seen in this group. After reading so many interesting viewpoints (maybe I missed some, due to the shear volume), I like to also contribute a tiny little bit to the original theme of Bruce Schneier.

In all fields of knowledge (including handcrafts) there are professionals and amateurs, the one group can't exist (by definition) without the other. There are two kinds of people, those who like to be professionals and those who like to be amateurs, with those who like to be professionals albeit necessarily have to start in the status of amateurs (or more appropriately appretices.) Becoming professionals have certian essential benefits, financial as well as social. That's what attract people of the first group. But people of the second group envisage other (individually different) advantages (of being amateurs). To use an analogon, a grandma may hate to learn and work as the star cook of the best-known restaurant but instead prefer to see her grandchildren enjoy her simple country dishes. In sport, Olympic participants are amateurs, while big money and spectacles are resverved for distinguished names like Boris and Steffi. Thus, professionals and amateurs co-exist and, I believe, should be able to co-exist peacefully, with each group profiting from the existence of the other. (In crypto, without the professionals the amateurs would lack general orientation for their endeavor and without the amateurs the would-be professionals (apprentices) wouldn't find the weak cryptos, the cracking of which constitutes the credentials for their ascension.)

Bruce Schneier has described a route for a would-be professional to proceed from the current amatuer status to the future professional status. Though this may be argued to be not the single possible route, I am sure that he has shown the most common and proper route. In fact the only way, for example, to become the world champion of boxing is to knock down every other competitors. There is no reason why things should be different in cryptology. If a cryptologist cracks all reputedly hard cryptos and nobody cracks his, he is duly the master and deserves a tenure.

However, I think it is correct to say that not all practical applications need the strongest crypto, not to mention that the very concept of the strength of crypto is subject to debate. Most secrets need not be kept very long, whether civil, military or political. On the other hand really unbreakable ciphers exist only in theory, if I don't err. Hence there is a wide spectrum of encryption algorithms conceivable, some strong, others weak, yet all applicable provided that they are used under the appropriate circumstances. Not always is the strongest crypto indicated. The best crypto may be unavailable due to patents, high cost, export restriction and crypto regulations, etc. etc. In such cases one has to look for comparatively weak cryptos.

With possible rare exceptions, amateurs can't compete with professionals. This is true in all fields. It follows that the design of the strongest ciphers is in a sense reserverd for the professional cryptologists. But that certainly doesn't preclude amateurs bringing forth good ciphers or even novel ideas. The critical issue is how this can happen in as favourable a manner as possible.

Being an amateur (a very humble one, due to my poor knowledge in the field) and on the assumption that the majority of participants in this group are amateurs (at least in the sense of Bruce Schneier), I venture to make a few suggestions that could be useful.

1. Often discussions in the group are less subject-oriented but carry a certain portion of sentiments. This is common in almost all internet groups I know of. However, this widening of the bandwidth tends to render the material less interesting, perhaps even boring, for the professionals, with the consequence that they wouldn't subscribe to the group and we have thus less chance to get valuable comments and critiques from them. Hence I like to suggest that general attention be paid to argue sharply and unambiguously without 'side-tracking' etc.

2. It appears that materials (documents) presented are often either difficult to understand or very incomplete (lacking details). This is at least my personal impression in trying recently to learn from two algorithms by authors of this group. A better documentation would facilitate the exchange of ideas, promote the spread of knowledge and thus further the progress of the group as a whole.

3. In the modern world a single person has only little chance of achieving very much. Collaboration is on the other hand highly effective in obtaining success. Dianelos mentioned recently that he once intended to initiate a collective design in our group of a cipher starting from an idea of him. I believe that our group has enough potential to indeed successfully carry out such projects, provided that these are appropriately managed in some way. Eventually the heatly debated opinions of the professionals on

amateurs as exemplified by the Memo of Bruce Schneier could get
modified.

I have yet some other thoughts. However, since these are related to
or in the same direction as the above, I believe it's better that I
cut short and await discussions (or flames).

M. K. Shen

M. K. Shen, Postfach 340238, D-80099 Muenchen, Germany
+49 (89) 831939 (6:00 GMT)
mok-kong.shen@stud.uni-muenchen.de
http://www.stud.uni-muenchen.de/~mok-kong.shen/

The words of a man's mouth are as deep waters, and the wellspring
of wisdom as a flowing brook. (Proverbs 18:4)

A little that a righteous man hath is better than the riches of many
wicked. (Psalms 37:16)

Subject: Re: Memo to the Amateur Cipher Designer
Date: Tue, 27 Oct 1998 21:02:07 GMT
From: schneier@counterpane.com (Bruce Schneier)
Message-ID: <36363316.8304318@news.visi.com>
References: <3635ED22.2F970FC3@stud.uni-muenchen.de>
Newsgroups: sci.crypt
Lines: 69

On Tue, 27 Oct 1998 16:56:18 +0100, Mok-Kong Shen
<mok-kong.shen@stud.uni-muenchen.de> wrote:

I agree with your distinction between amateurs and professionals, and
agree that there is room for both in cryptography. I don't think,
though, that my comments only applied to those who wanted to become
professionals. They applied to those who wanted to become good.
Whether they choose cryptography as a vocation or an avocation is not
particular relevent.

>However, I think it is correct to say that not all practical
>applications need the strongest crypto, not to mention that the very
>concept of the strength of crypto is subject to debate. Most secrets
>need not be kept very long, whether civil, military or political. On
>the other hand really unbreakable ciphers exist only in theory, if I
>don't err. Hence there is a wide spectrum of encryption algorithms
>conceivable, some strong, others weak, yet all applicable provided
>that they are used under the appropriate circumstances. Not always
>is the strongest crypto indicated. The best crypto may be unavailable
>due to patents, high cost, export restriction and crypto regulations,
>etc. etc. In such cases one has to look for comparatively weak cryptos.

While it is true that not every application need strong cryptography,
this does not mean that these applications should look towards weak
cryptography. Unlike physical locks on physical doors, weaker
cryptographic algorithms are not cheaper. They are not not faster,
don't take up less code, don't use less RAM, etc. There are certainly
exceptions--the identity cipher being the most flagrant example--but
in general strong cryptography is no more expensive than weak
cryptogreaphy. Hence, it makes sense to use the strongest
cryptography possible, regardless of the threat model.

>With possible rare exceptions, amateurs can't compete with
>professionals. This is true in all fields. It follows that the design
>of the strongest ciphers is in a sense reserverd for the professional
>cryptologists. But that certainly doesn't preclude amateurs bringing
>forth good ciphers or even novel ideas. The critical issue is how this
>can happen in as favourable a manner as possible.

I think cryptography is one of the few branches of mathematics where
the amateur can definitely compete with the professional. The field
is so new that anyone can learn the literature and contribute. There
are so many conferences and workshops that there are places for any
quality piece of research. There are a lot of amateurs out there
doing cryptography research, and many graduate students in
cryptography started out that way.

>3. In the modern world a single person has only little chance of
>   achieving very much. Collaboration is on the other hand highly
>   effective in obtaining success. Dianelos mentioned recently that
>   he once intended to initiate a collective design in our group of
>   a cipher starting from an idea of him. I believe that our group
>   has enough potential to indeed successfully carry out such projects,
>   provided that these are appropriately managed in some way.
>   Eventually the heatly debated opinions of the professionals on
>   amateurs as exemplified by the Memo of Bruce Schneier could get
>   modified.

This is an interesting thought. I don't believe a collaborative
design process would work at all--it's just too easy to propose ideas
without really knowing how good they are--a collaborative
cryptanalysis could be very interesting. Is there an interest in
finding an algorithm and, as a group, cryptanalyzing it?

Bruce
****
Bruce Schneier, President, Counterpane Systems Phone: 612-823-1098
101 E Minnehaha Parkway, Minneapolis, MN 55419 Fax: 612-823-1590
Free crypto newsletter. See: http://www.counterpane.com

Subject: Re: Memo to the Amateur Cipher Designer
Date: Tue, 27 Oct 1998 20:56:40 -0600
From: jgfunj@EnqvbSerrGrknf.pbz (W T Shaw)
Message-ID: <jgfunj-2710982056400001@dialup146.itexas.net>
References: <36363316.8304318@news.visi.com>
Newsgroups: sci.crypt
Lines: 19

In article <36363316.8304318@news.visi.com>, schneier@counterpane.com
(Bruce Schneier) wrote:
>
> I think cryptography is one of the few branches of mathematics where
> the amateur can definitely compete with the professional. The field
> is so new that anyone can learn the literature and contribute. There
> are so many conferences and workshops that there are places for any
> quality piece of research. There are a lot of amateurs out there
> doing cryptography research, and many graduate students in
> cryptography started out that way.

This means more than the *Memo* you posted. What you said above suggests
the importance of diversity of method and manner which is opposed to the
message of the Memo.
--

Please excuse if there are multiple postings for my responses...I have no idea where
they come from as I only receive one confirmation for each posting from my
newsserver.

Decrypt with ROT13 to get correct email address.

Subject: Re: Memo to the Amateur Cipher Designer
Date: Tue, 27 Oct 1998 21:04:20 -0600
From: jgfunj@EnqvbSerrGrknf.pbz (W T Shaw)
Message-ID: <jgfunj-2710982104200001@dialup146.itexas.net>
References: <36363316.8304318@news.visi.com>
Newsgroups: sci.crypt
Lines: 41

In article <36363316.8304318@news.visi.com>, schneier@counterpane.com
(Bruce Schneier) wrote:

> On Tue, 27 Oct 1998 16:56:18 +0100, Mok-Kong Shen
> <mok-kong.shen@stud.uni-muenchen.de> wrote:
>
> .... I think it is correct to say that not all practical
> >applications need the strongest crypto, not to mention that the very
> >concept of the strength of crypto is subject to debate. Most secrets
> >need not be kept very long, whether civil, military or political. On
> >the other hand really unbreakable ciphers exist only in theory, if I
> >don't err. Hence there is a wide spectrum of encryption algorithms
> >conceivable, some strong, others weak, yet all applicable provided
> >that they are used under the appropriate circumstances. Not always
> >is the strongest crypto indicated. The best crypto may be unavailable
> >due to patents, high cost, export restriction and crypto regulations,
> >etc. etc. In such cases one has to look for comparatively weak cryptos.
>
> While it is true that not every application need strong cryptography,
> this does not mean that these applications should look towards weak
> cryptography. Unlike physical locks on physical doors, weaker
> cryptographic algorithms are not cheaper. They are not not faster,
> don't take up less code, don't use less RAM, etc. There are certainly
> exceptions--the identity cipher being the most flagrant example--but
> in general strong cryptography is no more expensive than weak
> cryptogreaphy. Hence, it makes sense to use the strongest
> cryptography possible, regardless of the threat model.

I agree that strong crypto is desirable, but how you get there is most
important. Experience with a weaker version of an algorithm can teach you
many things. If true scalable algorithms are involved, it remains the
question of how strong do you want some implementation to be, always being
able to make it infinitely stronger.

There might be a twilight zone between weak and strong with a scalable
algorithm, it all depends on how you define these terms.
--

Please excuse if there are multiple postings for my responses...I have no idea where
they come from as I only receive one confirmation for each posting from my
newsserver.

Decrypt with ROT13 to get correct email address.

Subject: Re: Memo to the Amateur Cipher Designer

Date: Wed, 28 Oct 1998 11:59:03 +0100
From: Mok-Kong Shen <mok-kong.shen@stud.uni-muenchen.de>
Message-ID: <3636F8F7.F8287D6B@stud.uni-muenchen.de>
References: <36363316.8304318@news.visi.com>
Newsgroups: sci.crypt
Lines: 53

Bruce Schneier wrote:
>
> On Tue, 27 Oct 1998 16:56:18 +0100, Mok-Kong Shen
> <mok-kong.shen@stud.uni-muenchen.de> wrote:
  .....

>
> I agree with your distinction between amateurs and professionals, and
> agree that there is room for both in cryptography. I don't think,
> though, that my comments only applied to those who wanted to become
> professionals. They applied to those who wanted to become good.
> Whether they choose cryptography as a vocation or an avocation is not
> particular relevent.

Good knowledge of techniques of analysis is certainly indispensable
and I suppose everyone knows that but your Memo seems to imply that
no one should publish anything before he publishes successful analysis
of some (presumably good, known) algorithms. Now such algorithms are
limited in number. The easier jobs have probably already all been
discovered by the more capable professionals and done earlier, leaving
the newcommers little chance. Thus I think the requirement of proving
ones 'better' analysis capability is suppressive for novel design
ideas from coming up.

>
> >However, I think it is correct to say that not all practical
> >applications need the strongest crypto, not to mention that the very
> >concept of the strength of crypto is subject to debate. Most secrets
> >need not be kept very long, whether civil, military or political. On
> >the other hand really unbreakable ciphers exist only in theory, if I
> >don't err. Hence there is a wide spectrum of encryption algorithms
> >conceivable, some strong, others weak, yet all applicable provided
> >that they are used under the appropriate circumstances. Not always
> >is the strongest crypto indicated. The best crypto may be unavailable
> >due to patents, high cost, export restriction and crypto regulations,
> >etc. etc. In such cases one has to look for comparatively weak cryptos.

>
> While it is true that not every application need strong cryptography,
> this does not mean that these applications should look towards weak
> cryptography. Unlike physical locks on physical doors, weaker
> cryptographic algorithms are not cheaper. They are not not faster,
> don't take up less code, don't use less RAM, etc. There are certainly
> exceptions--the identity cipher being the most flagrant example--but
> in general strong cryptography is no more expensive than weak
> cryptogreaphy. Hence, it makes sense to use the strongest
> cryptography possible, regardless of the threat model.

Maybe I misunderstood you. But I don't see essential points of
disagreement between us in this respect. (Compare our two last
sentences.)

M. K. Shen

Subject: Re: Memo to the Amateur Cipher Designer
Date: Wed, 28 Oct 1998 15:32:00 GMT
From: schneier@counterpane.com (Bruce Schneier)
Message-ID: <36373706.886670@news.visi.com>
References: <3636F8F7.F8287D6B@stud.uni-muenchen.de>
Newsgroups: sci.crypt
Lines: 53

On Wed, 28 Oct 1998 11:59:03 +0100, Mok-Kong Shen
<mok-kong.shen@stud.uni-muenchen.de> wrote:

>Good knowledge of techniques of analysis is certainly indispensable
>and I suppose everyone knows that but your Memo seems to imply that
>no one should publish anything before he publishes successful analysis
>of some (presumably good, known) algorithms. Now such algorithms are
>limited in number. The easier jobs have probably already all been
>discovered by the more capable professionals and done earlier, leaving
>the newcommers little chance. Thus I think the requirement of proving
>ones 'better' analysis capability is suppressive for novel design
>ideas from coming up.

And I meant my memo to imply that: people who have not demonstrated
their ability to break algorithms are unlikely to develop algorithms
that cannot easily be broken. I don't believe the easier jobs havae
all been taken. Two designs from FSE 97 were easily broken in FSE 98.
Three AES designs were easily broken, and there have been small
weaknesses found in a few others. There are designs posted on
sci.crypt regularly that can be broken without developing any new
cryptanalytic techniques. In my "Self-Study Course" I listed some
algorithms that no one has bothered analyzing yet. There are
commercial designs--all the digital cellular algorithms, the Firewire
algorithms, etc--that should be looked at. There are Ritter's

designs. Any of these algorithms could potentially be cryptanalyzed
by amateurs. The easier jobs are not all taken, precisely becuase
there are so many of them.

>> >The best crypto may be unavailable
>> >due to patents, high cost, export restriction and crypto regulations,
>> >etc. etc. In such cases one has to look for comparatively weak cryptos.
>
>>There are certainly
>> exceptions--the identity cipher being the most flagrant example--but
>> in general strong cryptography is no more expensive than weak
>> cryptogreaphy. Hence, it makes sense to use the strongest
>> cryptography possible, regardless of the threat model.
>
>Maybe I misunderstood you. But I don't see essential points of
>disagreement between us in this respect. (Compare our two last
>sentences.)

If you are going to deliberately weaken an algorithm, fix some key
bits. Don't choose a random untested algorithm; you won't know how
strong or weak it is. And since there are a ready supply of tested,
trusted, unpatented, and free algorithms, I don't see this being much
of a problem.

Bruce
****
Bruce Schneier, President, Counterpane Systems   Phone: 612-823-1098
101 E Minnehaha Parkway, Minneapolis, MN 55419   Fax: 612-823-1590
Free crypto newsletter. See: http://www.counterpane.com

Subject: Re: Memo to the Amateur Cipher Designer
Date: Wed, 28 Oct 1998 17:31:59 +0100
From: Mok-Kong Shen <mok-kong.shen@stud.uni-muenchen.de>
Message-ID: <363746FF.B2E33D31@stud.uni-muenchen.de>
References: <36373706.886670@news.visi.com>
Newsgroups: sci.crypt
Lines: 43

Bruce Schneier wrote:
>

> algorithms, etc--that should be looked at. There are Ritter's
> designs. Any of these algorithms could potentially be cryptanalyzed
> by amateurs. The easier jobs are not all taken, precisely becuase
> there are so many of them.

I still guess that your logical argument is probably not perfect.
These are so to say 'ready foods' for the would-be professionals on
the way to their true professional status. Why have these been so
rarely attacked? Or are there barely any would-be professionals
around perhaps?

>
> >> >The best crypto may be unavailable
> >> >due to patents, high cost, export restriction and crypto regulations,
> >> >etc. etc. In such cases one has to look for comparatively weak cryptos.
> >
> >>There are certainly
> >> exceptions--the identity cipher being the most flagrant example--but
> >> in general strong cryptography is no more expensive than weak
> >> cryptogreaphy. Hence, it makes sense to use the strongest
> >> cryptography possible, regardless of the threat model.
> >
> >Maybe I misunderstood you. But I don't see essential points of
> >disagreement between us in this respect. (Compare our two last
> >sentences.)
>
> If you are going to deliberately weaken an algorithm, fix some key
> bits. Don't choose a random untested algorithm; you won't know how
> strong or weak it is. And since there are a ready supply of tested,
> trusted, unpatented, and free algorithms, I don't see this being much
> of a problem.

I said if the best crypto is unavailable than one has (is forced)
to take a weaker one. This does not imply one deliberately takes
the weakest of the available ones (only a fool would do that). You
said that one uses the strongest possible, i.e. the strongest of the
set of available ones. So there is no conflict between our opinions,
isn't it?

M. K. Shen

Subject: Re: Memo to the Amateur Cipher Designer
Date: Wed, 28 Oct 1998 22:41:04 GMT
From: schneier@counterpane.com (Bruce Schneier)
Message-ID: <36379ca4.2650087@news.visi.com>
References: <363746FF.B2E33D31@stud.uni-muenchen.de>
Newsgroups: sci.crypt
Lines: 46

On Wed, 28 Oct 1998 17:31:59 +0100, Mok-Kong Shen
<mok-kong.shen@stud.uni-muenchen.de> wrote:

>Bruce Schneier wrote:
>>
>
>> algorithms, etc--that should be looked at. There are Ritter's
>> designs. Any of these algorithms could potentially be cryptanalyzed
>> by amateurs. The easier jobs are not all taken, precisely becuase
>> there are so many of them.
>
>I still guess that your logical argument is probably not perfect.
>These are so to say 'ready foods' for the would-be professionals on
>the way to their true professional status. Why have these been so
>rarely attacked? Or are there barely any would-be professionals
>around perhaps?

Because people are busy. Because not everyone has time to spend weeks
(or days or even hours) analyzing every random cipher that comes
across their desk. Because the designs are not pubished, so the
breaks are not publishable. Beucause they are not widely known.
Because breaking them requires no new insights and hence is
uninteresting. For as many reasons as there are ciphers.

The argument "it's been around since 19xx and has not been broken,
therefor it is secure" is a flawed one. It assumes that people have
analyzed it during that time. Most ciphers are not analyzed by anyone
but the designers. This is why random designs are risky. And this is
also a great opportunity for someone who wants to learn. Cryptography
is rare, and possibly unique, in that a beginner can generate new--and
possibly publishable--results right from the beginning.

>I said if the best crypto is unavailable than one has (is forced)
>to take a weaker one. This does not imply one deliberately takes
>the weakest of the available ones (only a fool would do that). You
>said that one uses the strongest possible, i.e. the strongest of the
>set of available ones. So there is no conflict between our opinions,
>isn't it?

Don't think so.

Bruce
****
Bruce Schneier, President, Counterpane Systems Phone: 612-823-1098
101 E Minnehaha Parkway, Minneapolis, MN 55419 Fax: 612-823-1590
Free crypto newsletter. See: http://www.counterpane.com

Subject: Re: Memo to the Amateur Cipher Designer
Date: Thu, 29 Oct 1998 10:05:03 +0100
From: Mok-Kong Shen <mok-kong.shen@stud.uni-muenchen.de>
Message-ID: <36382FBF.7209447B@stud.uni-muenchen.de>
References: <36379ca4.2650087@news.visi.com>
Newsgroups: sci.crypt
Lines: 117

Bruce Schneier wrote:
>
> On Wed, 28 Oct 1998 17:31:59 +0100, Mok-Kong Shen
> <mok-kong.shen@stud.uni-muenchen.de> wrote:
>
> >Bruce Schneier wrote:
> >>
> >> algorithms, etc--that should be looked at. There are Ritter's
> >> designs. Any of these algorithms could potentially be cryptanalyzed
> >> by amateurs. The easier jobs are not all taken, precisely becuase
> >> there are so many of them.
> >>
> >I still guess that your logical argument is probably not perfect.
> >These are so to say 'ready foods' for the would-be professionals on
> >the way to their true professional status. Why have these been so
> >rarely attacked? Or are there barely any would-be professionals
> >around perhaps?
>
> Because people are busy. Because not everyone has time to spend weeks
> (or days or even hours) analyzing every random cipher that comes
> across their desk. Because the designs are not pubished, so the
> breaks are not publishable. Beucause they are not widely known.
> Because breaking them requires no new insights and hence is
> uninteresting. For as many reasons as there are ciphers.

I disagree. The would-be professionals are busy in attempting to
proving their 'better' (than his colleagues and certainly the
amateurs) analyis capability through cracking algorithms that are
presumably hard. They have thus strong incentives to do that work
which according to your Memo is sort of 'must'. Now it is also
my opinion that a number of algorithms published by amateurs are
difficult to understand (read) or very incomplete (lacking details)
(see a previous post of mine) or even obscure or trivial (your
'breaking requiring no new insights and hence uninteresting'). But I
would personally make an (at least one single) exception of
Terry Ritter's designs which you explicitly mentioned. Independent
of how easy or hard his designs can be broken, he has got patents.
Now it may well be argued whether obtaining pattens really means very
much. However a would-be professional choosing to break his designs
has an obvious advantage over breaking other equally weak (or harder)

algorithms. He could show off and say 'Hay, Look! I have cracked a couple of patented cryptos!' I can't imagine that such an advantange could be overlooked by any would-be professionals. Further, Ritter's work is apparently known to you to some degree. I believe that there are quite a number of the would-be professionals researching under your supervision and that you have very probably given to one or some of them a tip to attack Ritter's designs. A success in that would provide at least one very valuable 'insight' for general users of cryptological algorithms (and for the cryptology community as well), namely that the carrying of patents of cryptological algorithms is a very questionalbe qualification of the same and that these should be regarded with extreme care (suspicion) in evaluations. (Note: patents are published in government announcements. Scientific patents have at least the status of papers in established scientific journals, in particular can be assumed to have the same degree of 'known-ness' to researchers in the corresponding fields.)

>
> The argument "it's been around since 19xx and has not been broken,
> therefor it is secure" is a flawed one. It assumes that people have
> analyzed it during that time. Most ciphers are not analyzed by anyone
> but the designers. This is why random designs are risky. And this is
> also a great opportunity for someone who wants to learn. Cryptography
> is rare, and possibly unique, in that a beginner can generate new--and
> possibly publishable--results right from the beginning.

I wholly agree with you. Let me however remark that this is all very well known to this group. It appears time and again and repeatedly in posts of this group (I admit that sometimes I even found this theme boring) and has been well accepted and acknowleged to my knowledge. There is presently one exception, though, namely your sentence 'this is also a great opportunity for someone who wants to learn'. Do the would-be professionals (at least the beginners among them who have not yet accumulated too much knowledge) not want to learn? If yes, then there appears to be in my opinion a certain contradiction to what you wrote in the previous paragraph.

>
> >I said if the best crypto is unavailable than one has (is forced)
> >to take a weaker one. This does not imply one deliberately takes
> >the weakest of the available ones (only a fool would do that). You
> >said that one uses the strongest possible, i.e. the strongest of the
> >set of available ones. So there is no conflict between our opinions,
> >isn't it?
>
> Don't think so.

Please be kind enough to explain with a couple of sentences rather than making a difficult to comprehend categorical statement.

M. K. Shen

M. K. Shen, Postfach 340238, D-80099 Muenchen, Germany
+49 (89) 831939 (6:00 GMT)
mok-kong.shen@stud.uni-muenchen.de
http://www.stud.uni-muenchen.de/~mok-kong.shen/ (Last updated: 10th October 1998. origin site of WEAK1, WEAK2, WEAK3 and WEAK3-E. Containing 2 mathematical problems with rewards totalling US$500.)

Apply not techniques that you haven't fully understood. Use only subprograms that you have thoroughly verified. Never blindly trust what your colleagues claim. (a programmer advising novices, ~1970)

Sunshine is the best disinfectant.
(citation of a citation in B. Schneier and D. Banisar, The Electronic Privacy Papers. John-Wiley, New York, 1997.)

The words of a man's mouth are as deep waters, and the wellspring of wisdom as a flowing brook. (Proverbs 18:4)

A little that a righteous man hath is better than the riches of many wicked. (Psalms 37:16)

Subject: Re: Memo to the Amateur Cipher Designer
Date: Thu, 29 Oct 1998 15:09:50 GMT
From: schneier@counterpane.com (Bruce Schneier)
Message-ID: <363881dd.804381@news.visi.com>
References: <36382FBF.7209447B@stud.uni-muenchen.de>
Newsgroups: sci.crypt
Lines: 129

On Thu, 29 Oct 1998 10:05:03 +0100, Mok-Kong Shen
<mok-kong.shen@stud.uni-muenchen.de> wrote:

>Bruce Schneier wrote:
>>
>> On Wed, 28 Oct 1998 17:31:59 +0100, Mok-Kong Shen
>> <mok-kong.shen@stud.uni-muenchen.de> wrote:

```
>>
>> >Bruce Schneier wrote:
>> >>
>> >> algorithms, etc--that should be looked at.  There are Ritter's
>> >> designs.  Any of these algorithms could potentially be cryptanalyzed
>> >> by amateurs.  The easier jobs are not all taken, precisely becuase
>> >> there are so many of them.
>> >
>> >I still guess that your logical argument is probably not perfect.
>> >These are so to say 'ready foods' for the would-be professionals on
>> >the way to their true professional status. Why have these been so
>> >rarely attacked? Or are there barely any would-be professionals
>> >around perhaps?
>>
>> Because people are busy.  Because not everyone has time to spend weeks
>> (or days or even hours) analyzing every random cipher that comes
>> across their desk.  Because the designs are not pubished, so the
>> breaks are not publishable.  Beucause they are not widely known.
>> Because breaking them requires no new insights and hence is
>> uninteresting.  For as many reasons as there are ciphers.
>
>I disagree. The would-be professionals are busy in attempting to
>proving their 'better' (than his colleagues and certainly the
>amateurs) analyis capability through cracking algorithms that are
>presumably hard. They have thus strong incentives to do that work
>which according to your Memo is sort of 'must'. Now it is also
>my opinion that a number of algorithms published by amateurs are
>difficult to understand (read) or very incomplete (lacking details)
>(see a previous post of mine) or even obscure or trivial (your
>'breaking requiring no new insights and hence uninteresting'). But I
>would personally make an (at least one single) exception of
>Terry Ritter's designs which you explicitly mentioned. Independent
>of how easy or hard his designs can be broken, he has got patents.
>Now it may well be argued whether obtaining pattens really means very
>much. However a would-be professional choosing to break his designs
>has an obvious advantage over breaking other equally weak (or harder)
>algorithms. He could show off and say 'Hay, Look! I have cracked a
>couple of patented cryptos!' I can't imagine that such an advantange
>could be overlooked by any would-be professionals. Further, Ritter's
>work is apparently known to you to some degree. I believe that there
>are quite a number of the would-be professionals researching under
>your supervision and that you have very probably given to one or some
>of them a tip to attack Ritter's designs. A success in that would
>provide at least one very valuable 'insight' for general users of
>cryptological algorithms (and for the cryptology community as well),
>namely that the carrying of patents of cryptological algorithms is
>a very questionalbe qualification of the same and that these should
>be regarded with extreme care (suspicion) in evaluations. (Note:
>patents are published in government announcements. Scientific patents
>have at least the status of papers in established scientific journals,
>in particular can be assumed to have the same degree of 'known-ness'
>to researchers in the corresponding fields.)

I don't understand.  Do you disagree with reality (that there are all
these ciphers that are not being looked at) or with my reasoning as to
why they are not being looked at?  I don't know what to tell you.  I
know all of the algorithms I listed in my previous posting have not
been looked at by the academic cryptographers who I think of as the
"good cryptanalysts."  I know the reasons listed are ones that I have
heard others use or use myself.  Maybe you're right--these algorithms
have been analyzed and some of them have been broken--and the breaks
have either not been published or have been published in places I dont
know about, but I kind of doubt that.

Many of us have breaks of amateur ciphers, ones that appear on
sc.crypt, get patents, or are used opterationally, that we just don't
have time to write up or flesh out.  It's just not worth the bother.

I don't mean this to be statement of opinion, but a statement of fact.
Fact 1:  There are many unpublished, and even some published ones,
that no one has bothered trying to cryptanalyze.  Fact 2:  Some of the
reasons people give for not bothering are listed above.

>> The argument "it's been around since 19xx and has not been broken,
>> therefor it is secure" is a flawed one.  It assumes that people have
>> analyzed it during that time.  Most ciphers are not analyzed by anyone
>> but the designers.  This is why random designs are risky.  And this is
>> also a great opportunity for someone who wants to learn.  Cryptography
>> is rare, and possibly unique, in that a beginner can generate new--and
>> possibly publishable--results right from the beginning.
>
>I wholly agree with you. Let me however remark that this is all
>very well known to this group. It appears time and again and repeatedly
>in posts of this group (I admit that sometimes I even found this theme
>boring) and has been well accepted and acknowleged to my knowledge.
>There is presently one exception, though, namely your sentence 'this
>is also a great opportunity for someone who wants to learn'. Do the
>would-be professionals (at least the beginners among them who have not
>yet accumulated too much knowledge) not want to learn? If yes, then
>there appears to be in my opinion a certain contradiction to what you
>wrote in the previous paragraph.

I believe that:  1) There are very few beginner cryptanalysts.  2)
They tend to try to reproduce published results, as I described in my
"Self Study Course in Block Cipher Cryptanalysis.  3) They don't know
```

about the random designs that appear. (Remember, most people in the field don't EVER read sci.crypt.) 4) Some realize that they need to break things to learn.

>> >I said if the best crypto is unavailable than one has (is forced)
>> >to take a weaker one. This does not imply one deliberately takes
>> >the weakest of the available ones (only a fool would do that). You
>> >said that one uses the strongest possible, i.e. the strongest of the
>> >set of available ones. So there is no conflict between our opinions,
>> >isn't it?
>>
>> Don't think so.
>
>Please be kind enough to explain with a couple of sentences rather
>than making a difficult to comprehend categorical statement.

I do not believe there is any conflict between our opinions. I believe that your opinion and mine are not in conflict, meaning that they can coexist without conflict, but strongly implying (and I mean this too) that they are compatible and in agreement. (Honestly, I don't know how else to explain it.)

Bruce

****
Bruce Schneier, President, Counterpane Systems Phone: 612-823-1098
101 E Minnehaha Parkway, Minneapolis, MN 55419 Fax: 612-823-1590
Free crypto newsletter. See: http://www.counterpane.com

Subject: Re: Memo to the Amateur Cipher Designer
Date: Thu, 29 Oct 1998 16:58:33 +0100
From: Mok-Kong Shen <mok-kong.shen@stud.uni-muenchen.de>
Message-ID: <363890A9.605EB4BF@stud.uni-muenchen.de>
References: <363881dd.804381@news.visi.com>
Newsgroups: sci.crypt
Lines: 114

Bruce Schneier wrote:
>
> On Thu, 29 Oct 1998 10:05:03 +0100, Mok-Kong Shen
> <mok-kong.shen@stud.uni-muenchen.de> wrote:
>
> >Bruce Schneier wrote:
> >>
> >> Because people are busy. Because not everyone has time to spend weeks
> >> (or days or even hours) analyzing every random cipher that comes
> >> across their desk. Because the designs are not pubished, so the
> >> breaks are not publishable. Beucause they are not widely known.
> >> Because breaking them requires no new insights and hence is
> >> uninteresting. For as many reasons as there are ciphers.
> >>
> >I disagree. The would-be professionals are busy in attempting to
> >proving their 'better' (than his colleagues and certainly the
> >amateurs) analyis capability through cracking algorithms that are
> >presumably hard. They have thus strong incentives to do that work
> >which according to your Memo is sort of 'must'. Now it is also
> >my opinion that a number of algorithms published by amateurs are
> >difficult to understand (read) or very incomplete (lacking details)
> >(see a previous post of mine) or even obscure or trivial (your
> >'breaking requiring no new insights and hence uninteresting'). But I
> >would personally make an (at least one single) exception of
> >Terry Ritter's designs which you explicitly mentioned. Independent
> >of how easy or hard his designs can be broken, he has got patents.
> >Now it may well be argued whether obtaining pattens really means very
> >much. However a would-be professional choosing to break his designs
> >has an obvious advantage over breaking other equally weak (or harder)
> >algorithms. He could show off and say 'Hay, Look! I have cracked a
> >couple of patented cryptos!' I can't imagine that such an advantange
> >could be overlooked by any would-be professionals. Further, Ritter's
> >work is apparently known to you to some degree. I believe that there
> >are quite a number of the would-be professionals researching under
> >your supervision and that you have very probably given to one or some
> >of them a tip to attack Ritter's designs. A success in that would
> >provide at least one very valuable 'insight' for general users of
> >cryptological algorithms (and for the cryptology community as well),
> >namely that the carrying of patents of cryptological algorithms is
> >a very questionalbe qualification of the same and that these should
> >be regarded with extreme care (suspicion) in evaluations. (Note:
> >patents are published in government announcements. Scientific patents
> >have at least the status of papers in established scientific journals,
> >in particular can be assumed to have the same degree of 'known-ness'
> >to researchers in the corresponding fields.)
>
> I don't understand. Do you disagree with reality (that there are all
> these ciphers that are not being looked at) or with my reasoning as to
> why they are not being looked at? I don't know what to tell you. I
> know all of the algorithms I listed in my previous posting have not
> been looked at by the academic cryptographers who I think of as the
> "good cryptanalysts." I know the reasons listed are ones that I have
> heard others use or use myself. Maybe you're right--these algorithms
> have been analyzed and some of them have been broken--and the breaks
> have either not been published or have been published in places I dont
> know about, but I kind of doubt that.
>

> Many of us have breaks of amateur ciphers, ones that appear on
> sc.crypt, get patents, or are used opterationally, that we just don't
> have time to write up or flesh out. It's just not worth the bother.
>
> I don't mean this to be statement of opinion, but a statement of fact.
> Fact 1: There are many unpublished, and even some published ones,
> that no one has bothered trying to cryptanalyze. Fact 2: Some of the
> reasons people give for not bothering are listed above.

In response to your point 'I don't understand': You said that because people are busy no one has the time to look at the amateur ciphers that are unpublished, etc. etc. I argued, hopefully convincingly and clearly, that at least Terry Ritter's designs do deserve being analyzed by the would-be professionals and these should be (the or one of) their first choice (of amateur algorithms) of objects of attack. For to these every one of your arguments of 'not published' etc. etc. evidently do not apply. Since Ritter's work is not new at all, the cracks must have been successful if his designs were indeed so weak as your wordings clearly claimed them to be.

Let me perhaps quote what you wrote previously in this thread to illustrate the inconsistency of your logic, particularly in reference to Ritter's work. In response to Ritter's post of 20 Oct. 00:40:21 you wrote on 26 Oct 03:59:50 :

   I invite you to submit a paper, based on your patent #5,727,062
   ("Variable Size Block Ciphers") to the 1999 Fast Software Encryption
   workshop. I believe it will be published.

However on 28 Oct 15:32:00 you wrote

   There are Ritter's designs. Any of these algorithms could
   potentially be cryptanalyzed by amateurs.

Unless you are firmly of the opinion that FSE workshop is at such a (low) level such that it readily accepts papers presenting algorithms that could potentially be cryptanalyzed by amateurs (I am afraid the program committee of FES workshop would be angry with you) the above two paragraphs are entirely incompatible with each other in my humble view.

> >> Don't think so.
> >
> >Please be kind enough to explain with a couple of sentences rather
> >than making a difficult to comprehend categorical statement.
>
> I do not believe there is any conflict between our opinions. I
> believe that your opinion and mine are not in conflict, meaning that
> they can coexist without conflict, but strongly implying (and I mean
> this too) that they are compatible and in agreement. (Honestly, I
> don't know how else to explain it.)

Well, I think that does finally explain away the phrase in question.
Thanks.

M. K. Shen

Subject: Re: Memo to the Amateur Cipher Designer
Date: Thu, 29 Oct 1998 16:41:19 GMT
From: schneier@counterpane.com (Bruce Schneier)
Message-ID: <3638987e.6598604@news.visi.com>
References: <363890A9.605EB4BF@stud.uni-muenchen.de>
Newsgroups: sci.crypt
Lines: 70

On Thu, 29 Oct 1998 16:58:33 +0100, Mok-Kong Shen
<mok-kong.shen@stud.uni-muenchen.de> wrote:
>In response to your point 'I don't understand': You said that because
>people are busy no one has the time to look at the amateur ciphers
>that are unpublished, etc. etc. I argued, hopefully convincingly and
>clearly, that at least Terry Ritter's designs do deserve being
>analyzed by the would-be professionals and these should be (the or
>one of) their first choice (of amateur algorithms) of objects of
>attack. For to these every one of your arguments of 'not published'
>etc. etc. evidently do not apply. Since Ritter's work is not new at
>all, the cracks must have been successful if his designs were indeed
>so weak as your wordings clearly claimed them to be.

You know, I don't want to pick on Ritter in particular here. I don't know about whether his designs "deserve" to be analyzed; that is a value judgment. I don't know if they are strong or weak. I do believe that, as I said below, they "could potentially be be cryptanalyzed by amateurs." (Note the word "potentially." I put that in there to indicate that I did not know if they actually could.) I know that the cryptanalysts I have talked with have not looked at his designs at all (again, it's the "published in a refereed journal or conference" business--like it or not, it's a real issue in the publish-or-perish world of academia). Feel free to email any cryptanalyst you want and argue that his reasons are wrong; I cannot stop you. I can only state my reasons, and list reasons I have heard from others.

>Let me perhaps quote what you wrote previously in this thread to
>illustrate the inconsistency of your logic, particularly in reference
>to Ritter's work. In response to Ritter's post of 20 Oct. 00:40:21
>you wrote on 26 Oct 03:59:50 :
>
> I invite you to submit a paper, based on your patent #5,727,062
> ("Variable Size Block Ciphers") to the 1999 Fast Software Encryption
> workshop. I believe it will be published.
>
>However on 28 Oct 15:32:00 you wrote
>
> There are Ritter's designs. Any of these algorithms could
> potentially be cryptanalyzed by amateurs.
>
>Unless you are firmly of the opinion that FSE workshop is at such
>a (low) level such that it readily accepts papers presenting algorithms
>that could potentially be cryptanalyzed by amateurs (I am afraid the
>program committee of FES workshop would be angry with you) the
>above two paragraphs are entirely incompatible with each other in my
>humble view.

Being on the program committee of FSE, I can categorically state that the conference accepts papers preventing algorithms that could potentially be cryptanalyzed by amateurs. In FSE 97 we saw ICE and TwoPrime, both of which were easy cryptanalyses. In SAC 96 we saw Akelarre, which could have been cryptanalyzed by someone with not a lot of skill. I assure you, the program committee is angry with itself when a medeocre design slips in, but it happens. FSE is supposed to publish cipher designs, so that there is fodder to be analyzed.

Really, analysis isn't this big mysterious thing that only a few people can do so the rest of the world might as well not bother. It isn't true that there aren't things to analyze out there. It isn't true that all the "low hanging fruit" is taken. It is true that most of the "low hanging publishable fruit" is taken, but by no means all of it.

Bruce
****
Bruce Schneier, President, Counterpane Systems Phone: 612-823-1098
101 E Minnehaha Parkway, Minneapolis, MN 55419 Fax: 612-823-1590
Free crypto newsletter. See: http://www.counterpane.com

Subject: Re: Memo to the Amateur Cipher Designer
Date: Fri, 30 Oct 1998 00:29:36 GMT
From: dscott@networkusa.net
Message-ID: <71b19g$m61$1@nnrp1.dejanews.com>
References: <3638987e.6598604@news.visi.com>
Newsgroups: sci.crypt
Lines: 86

In article <3638987e.6598604@news.visi.com>,
  schneier@counterpane.com (Bruce Schneier) wrote:
> On Thu, 29 Oct 1998 16:58:33 +0100, Mok-Kong Shen
> <mok-kong.shen@stud.uni-muenchen.de> wrote:
> >In response to your point 'I don't understand': You said that because
> >people are busy no one has the time to look at the amateur ciphers
> >that are unpublished, etc. etc. I argued, hopefully convincingly and
> >clearly, that at least Terry Ritter's designs do deserve being
> >analyzed by the would-be professionals and these should be (the or
> >one of) their first choice (of amateur algorithms) of objects of
> >attack. For to these every one of your arguments of 'not published'
> >etc. etc. evidently do not apply. Since Ritter's work is not new at
> >all, the cracks must have been successful if his designs were indeed
> >so weak as your wordings clearly claimed them to be.
>
> You know, I don't want to pick on Ritter in particular here. I don't
> know about whether his designs "deserve" to be analyzed; that is a
> value judgment. I don't know if they are strong or weak. I do
> believe that, as I said below, they "could potentially be be
> cryptanalyzed by amateurs." (Note the word "potentially." I put that
> in there to indicate that I did not know if they actually could.) I
> know that the cryptanalysts I have talked with have not looked at his
> designs at all (again, it's the "published in a refereed journal or
> conference" business--like it or not, it's a real issue in the
> publish-or-perish world of academia). Feel free to email any
> cryptanalyst you want and argue that his reasons are wrong; I cannot
> stop you. I can only state my reasons, and list reasons I have heard
> from others.
>
> >Let me perhaps quote what you wrote previously in this thread to
> >illustrate the inconsistency of your logic, particularly in reference
> >to Ritter's work. In response to Ritter's post of 20 Oct. 00:40:21
> >you wrote on 26 Oct 03:59:50 :
> >
> > I invite you to submit a paper, based on your patent #5,727,062
> > ("Variable Size Block Ciphers") to the 1999 Fast Software Encryption
> > workshop. I believe it will be published.
> >
> >However on 28 Oct 15:32:00 you wrote
> >
> > There are Ritter's designs. Any of these algorithms could

> > potentially be cryptanalyzed by amateurs.
> >
> >Unless you are firmly of the opinion that FSE workshop is at such
> >a (low) level such that it readily accepts papers presenting algorithms
> >that could potentially be cryptanalyzed by amateurs (I am afraid the
> >program committee of FES workshop would be angry with you) the
> >above two paragraphs are entirely incompatible with each other in my
> >humble view.
>
> Being on the program committee of FSE, I can categorically state that
> the conference accepts papers preventing algorithms that could
> potentially be cryptanalyzed by amateurs. In FSE 97 we saw ICE and
> TwoPrime, both of which were easy cryptanalyses. In SAC 96 we saw
> Akelarre, which could have been cryptanalyzed by someone with not a
> lot of skill. I assure you, the program committee is angry with
> itself when a medeocre design slips in, but it happens. FSE is
> supposed to publish cipher designs, so that there is fodder to be
> analyzed.
>

Sorry mok-kong I agree with Bruce they allow some easy to brake ciphers in so that it can inflate there egos. But they really don't wish to try to look at any thing to difficult or out of the main stream since if they are like Bruce they lack the intelligence to anallize something not done over and over or in a all or nothing sort of way that there little pee brains can't conceive of.

So I guess I sometimes do agree with limited parts of what Bruce Babels out. Of course a lot of his babel covers both sides so like a good sleazy politican you can't disagree with every thing the guy says.

But that is just my humble opinion of pompous Bruce the Spammer. Don't expect to see a published real crack of scott19u since it is to hard for them.

--
http://cryptography.org/cgi-bin/crypto.cgi/Misc/scott19u.zip
ftp search for scott*u.zip in norway

----------== Posted via Deja News, The Discussion Network ==----------
http://www.dejanews.com/ Search, Read, Discuss, or Start Your Own

---

Subject: Re: Memo to the Amateur Cipher Designer
Date: Fri, 30 Oct 1998 11:18:36 +0100
From: Mok-Kong Shen <mok-kong.shen@stud.uni-muenchen.de>
Message-ID: <3639927C.D5B723BF@stud.uni-muenchen.de>
References: <71b19g$m61$1@nnrp1.dejanews.com>
Newsgroups: sci.crypt
Lines: 53

dscott@networkusa.net wrote:

> But that is just my humble opinion of pompous Bruce the
> Spammer. Don't expect to see a published real crack of scott19u
> since it is to hard for them.

Very sorry that I am not on your side. Quite a lot of what Bruce Schneier said does correspond to the reality (a sad reality though), even though I can't agree with him in a certain point and have shown recently (I hope clearly) his logical inconsistency there.

As long as some writings are not clearly written and the authors seem to be not motivated to improve them, there is no chance at all that these could get into publications (journals, official publications, etc.) and thus these would not be read, let alone recognized. One may invent a really good crypto algorithm, but if the description is messy, one can't blame anybody for being neglected. For the readers can't be expected to 'decrypt' obscure descriptions. Papers being chosen by an editorial committe are undergoing a real competition process. For two papers of equal scientific value, one readable the other not, it is only fair that the first one gets chosen. Those who have published papers have a certain advantage over the newcomers in that they know well from experience how to write manuscripts in styles that maximize the chance of getting accepted. In other words the newcomers should take more effort to present their thoughts convincingly and write exceptionally clearly.

Returning to the general situation of amateur ciphers, I have expressed in a previous post my personal view that many (I hesitate to use a superlative word in fear of flames) are not described in a style common to publications in scientific literatures. (Lest there be misunderstandings, let me say that my own stuffs put on my Web page are no good in style and related aspects.) There is therefore a bad need for us amateurs to take effort in presenting our materials appropriately such that they are palatable to the professionals and the academia. Needless to say that a weak cipher with a super description is nothing. But the other way round, a good cipher with a poor description will find no user. It thus lastly all depends on us ourselves whether we like to have our designs used

by the public or just left as precious objects in our living rooms
for showing to visitors from time to time. I hope that in future
we all (I mean the amateurs among the subscribers of the group) could
cooperate (through reciprocal non-sentimental critiques and comments)
in such a way that we not only produce good cipher products but also
have excellent descriptions of the same to present them to the
professionals and the public. If on the other hand the current
situation remains unaltered, then I am very pessimistic. Having said
this, I suppose it is understandable that I personally believe that
the Memo of Bruce Schneier, all the heated debates notwithstanding,
probably has done for us something fairly positive and for that
contribution I like to express my appreciation.

M. K. Shen

Subject: Re: Memo to the Amateur Cipher Designer
Date: Fri, 30 Oct 1998 15:25:03 +0100
From: <tbb03ar@mail.lrz-muenchen.de>
Message-ID: <Pine.GSO.4.03.9810301420580.1419-100000@sun5.lrz-muenchen.de>
References: <3639927C.D5B723BF@stud.uni-muenchen.de>
Newsgroups: sci.crypt
Lines: 39

On Fri, 30 Oct 1998, Mok-Kong Shen wrote:

> ...
> There is therefore
> a bad need for us amateurs to take effort in presenting our materials
> appropriately such that they are palatable to the professionals
> and the academia. Needless to say that a weak cipher with a super
> description is nothing. But the other way round, a good cipher
> with a poor description will find no user.

I for my own prefer the weak cipher with good description, not to protect
data but to argue and to learn how to develope stronger ciphers.

> It thus lastly all
> depends on us ourselves whether we like to have our designs used
> by the public or just left as precious objects in our living rooms
> for showing to visitors from time to time. I hope that in future
> we all (I mean the amateurs among the subscribers of the group) could
> cooperate (through reciprocal non-sentimental critiques and comments)
> in such a way that we not only produce good cipher products but also
> have excellent descriptions of the same to present them to the
> professionals and the public.

I'd like to see a group that tries to develop and to break amateur
ciphers - not as a group of cryptographers that develope strong ciphers,
but as cryptanalyticers (something like the ACA but working with
computers and modern cryptanalysis).

The people within this group would be able to do better cryptanalysis and
would be more familiar with the language and descriptions used by
cryptographers. This way they would be able to test and publish ciphers
and to become accepted by the professionals.

Andreas Enterrottacher

enterrottacher@lrz.tu-muenchen.de
enterrottacher@t-online.de

Subject: Re: Memo to the Amateur Cipher Designer
Date: Fri, 30 Oct 1998 15:05:16 -0600
From: jgfunj@EnqvbSerrGrknf.pbz (W T Shaw)
Message-ID: <jgfunj-3010981505170001@dialup68.itexas.net>
References: <Pine.GSO.4.03.9810301420580.1419-100000@sun5.lrz-muenchen.de>
Newsgroups: sci.crypt
Lines: 30

In article <Pine.GSO.4.03.9810301420580.1419-100000@sun5.lrz-muenchen.de>,
<tbb03ar@mail.lrz-muenchen.de> wrote:
>
> I'd like to see a group that tries to develop and to break amateur
> ciphers - not as a group of cryptographers that develope strong ciphers,
> but as cryptanalyticers (something like the ACA but working with
> computers and modern cryptanalysis).

Many in the ACA are working with computers and extending their
capabilities. The first hurdle has been in developing automated means of
solving all ciphers in the ACA stable. While being generally successful,
a small minority of sequences require rather elaborate judgement to guess
their plaintext if it can be determined at all.

The future activity of the computer oriented segement of the group is
largely determined by suggestions and contributions. Can join the ACA for
a nominal annual fee, largely used mainly to cover current expenses. The
American Cryptogram Association has members throughout the world, so don't
let the historic name throw you.

This month's The Cryptogram marks the beginning of the formal use of

QBasic in the Computer Column. The column author states that code will be placed at the Crypto Drop Box, http://www.und.nodak.edu/org/crypto/crypto, which full of cryptographic resources.
--

Heard recently on Larry King: Jimmy Carter and Billy Graham agreeing that it is sometimes wise to tell a lie.

Decrypt with ROT13 to get correct email address.

Subject: Re: Memo to the Amateur Cipher Designer
Date: Sat, 31 Oct 1998 06:49:46 GMT
From: "Douglas A. Gwyn" <DAGwyn@null.net>
Message-ID: <363AB2AC.B6B67D8D@null.net>
References: <Pine.GSO.4.03.9810301420580.1419-100000@sun5.lrz-muenchen.de>
Newsgroups: sci.crypt
Lines: 8

tbb03ar@mail.lrz-muenchen.de wrote:
> I'd like to see a group that tries to develop and to break amateur
> ciphers - not as a group of cryptographers that develope strong ciphers,
> but as cryptanalyticers (something like the ACA but working with
> computers and modern cryptanalysis).

Actually, the ACA does have a section devoted to computers.
But it needs more members!

Subject: Re: Memo to the Amateur Cipher Designer
Date: Fri, 30 Oct 1998 09:56:24 +0100
From: Mok-Kong Shen <mok-kong.shen@stud.uni-muenchen.de>
Message-ID: <36397F38.A8381A2@stud.uni-muenchen.de>
References: <3638987e.6598604@news.visi.com>
Newsgroups: sci.crypt
Lines: 77

Bruce Schneier wrote:
>
> On Thu, 29 Oct 1998 16:58:33 +0100, Mok-Kong Shen
> <mok-kong.shen@stud.uni-muenchen.de> wrote:
> >In response to your point 'I don't understand': You said that because
> >people are busy no one has the time to look at the amateur ciphers
> >that are unpublished, etc. etc. I argued, hopefully convincingly and
> >clearly, that at least Terry Ritter's designs do deserve being
> >analyzed by the would-be professionals and these should be (the or
> >one of) their first choice (of amateur algorithms) of objects of
> >attack. For to these every one of your arguments of 'not published'
> >etc. etc. evidently do not apply. Since Ritter's work is not new at
> >all, the cracks must have been successful if his designs were indeed
> >so weak as your wordings clearly claimed them to be.
>
> You know, I don't want to pick on Ritter in particular here. I don't
> know about whether his designs "deserve" to be analyzed; that is a
> value judgment. I don't know if they are strong or weak. I do
> believe that, as I said below, they "could potentially be be
> cryptanalyzed by amateurs." (Note the word "potentially." I put that
> in there to indicate that I did not know if they actually could.) I
> know that the cryptanalysts I have talked with have not looked at his
> designs at all (again, it's the "published in a refereed journal or
> conference" business--like it or not, it's a real issue in the
> publish-or-perish world of academia). Feel free to email any
> cryptanalyst you want and argue that his reasons are wrong; I cannot
> stop you. I can only state my reasons, and list reasons I have heard
> from others.

My point was Ritter hat got patents. Patents are published by the governments. Certain very good algorithms have patents that prevent free use. Are you saying that the academia neglects the patent publications? Some crypto patents are claimed by highly distinguished figures of the academia! You said now you don't want to pick Ritter in particular. But you explicitly (and singularly) mentioned Ritter's designs in a particular context in a previous post!

>
> >Let me perhaps quote what you wrote previously in this thread to
> >illustrate the inconsistency of your logic, particularly in reference
> >to Ritter's work. In response to Ritter's post of 20 Oct. 00:40:21
> >you wrote on 26 Oct 03:59:50 :
> >
> > I invite you to submit a paper, based on your patent #5,727,062
> > ("Variable Size Block Ciphers") to the 1999 Fast Software Encryption
> > workshop. I believe it will be published.
> >
> >However on 28 Oct 15:32:00 you wrote
> >
> > There are Ritter's designs. Any of these algorithms could
> > potentially be cryptanalyzed by amateurs.
> >
> >Unless you are firmly of the opinion that FSE workshop is at such
> >a (low) level such that it readily accepts papers presenting algorithms
> >that could potentially be cryptanalyzed by amateurs (I am afraid the
> >program committee of FES workshop would be angry with you) the

> >above two paragraphs are entirely incompatible with each other in my
> >humble view.
>
> Being on the program committee of FSE, I can categorically state that
> the conference accepts papers preventing algorithms that could
> potentially be cryptanalyzed by amateurs. In FSE 97 we saw ICE and
> TwoPrime, both of which were easy cryptanalyses. In SAC 96 we saw
> Akelarre, which could have been cryptanalyzed by someone with not a
> lot of skill. I assure you, the program committee is angry with
> itself when a medeocre design slips in, but it happens. FSE is
> supposed to publish cipher designs, so that there is fodder to be
> analyzed.

I find it is surprising that someone on the program committee can
be so inconsiderate of the committee and make use of the name of the
workshop to argue for his own issues in logically entirely inconsitent
manner. Please say in clear terms if what I pointed out as
contradiction is false.

M. K. Shen

Subject: Re: Memo to the Amateur Cipher Designer
Date: Fri, 30 Oct 1998 12:31:20 GMT
From: dscott@networkusa.net
Message-ID: <71cbio$a4o$1@nnrp1.dejanews.com>
References: <36397F38.A8381A2@stud.uni-muenchen.de>
Newsgroups: sci.crypt
Lines: 25

In article <36397F38.A8381A2@stud.uni-muenchen.de>,
  Mok-Kong Shen <mok-kong.shen@stud.uni-muenchen.de> wrote:
> Bruce Schneier wrote:
> >...snip...

>
> I find it is surprising that someone on the program committee can
> be so inconsiderate of the committee and make use of the name of the
> workshop to argue for his own issues in logically entirely inconsitent
> manner. Please say in clear terms if what I pointed out as
> contradiction is false.
>
> M. K. Shen
>

  I guess then you greatly underestamate the EGO of the phony
crypto gods. Weak up have some gin sing or something.

--
http://cryptography.org/cgi-bin/crypto.cgi/Misc/scott19u.zip
ftp search for scott*u.zip in norway

----------== Posted via Deja News, The Discussion Network ==----------
http://www.dejanews.com/ Search, Read, Discuss, or Start Your Own

---

Subject: Re: Memo to the Amateur Cipher Designer
Date: Fri, 30 Oct 1998 15:31:06 GMT
From: schneier@counterpane.com (Bruce Schneier)
Message-ID: <3639d8bf.741333@news.visi.com>
References: <36397F38.A8381A2@stud.uni-muenchen.de>
Newsgroups: sci.crypt
Lines: 137

On Fri, 30 Oct 1998 09:56:24 +0100, Mok-Kong Shen
<mok-kong.shen@stud.uni-muenchen.de> wrote:

>Bruce Schneier wrote:
>>
>> On Thu, 29 Oct 1998 16:58:33 +0100, Mok-Kong Shen
>> <mok-kong.shen@stud.uni-muenchen.de> wrote:
>> >In response to your point 'I don't understand': You said that because
>> >people are busy no one has the time to look at the amateur ciphers
>> >that are unpublished, etc. etc. I argued, hopefully convincingly and
>> >clearly, that at least Terry Ritter's designs do deserve being
>> >analyzed by the would-be professionals and these should be (the or
>> >one of) their first choice (of amateur algorithms) of objects of
>> >attack. For to these every one of your arguments of 'not published'
>> >etc. etc. evidently do not apply. Since Ritter's work is not new at
>> >all, the cracks must have been successful if his designs were indeed
>> >so weak as your wordings clearly claimed them to be.
>>
>> You know, I don't want to pick on Ritter in particular here. I don't
>> know about whether his designs "deserve" to be analyzed; that is a
>> value judgment. I don't know if they are strong or weak. I do
>> believe that, as I said below, they "could potentially be be
>> cryptanalyzed by amateurs." (Note the word "potentially." I put that
>> in there to indicate that I did not know if they actually could.) I
>> know that the cryptanalysts I have talked with have not looked at his
>> designs at all (again, it's the "published in a refereed journal or
>> conference" business--like it or not, it's a real issue in the
>> publish-or-perish world of academia). Feel free to email any
>> cryptanalyst you want and argue that his reasons are wrong; I cannot

>> stop you. I can only state my reasons, and list reasons I have heard
>> from others.
>
>My point was Ritter hat got patents. Patents are published by the
>governments. Certain very good algorithms have patents that prevent
>free use. Are you saying that the academia neglects the patent
>publications? Some crypto patents are claimed by highly distinguished
>figures of the academia! You said now you don't want to pick Ritter in
>particular. But you explicitly (and singularly) mentioned Ritter's
>designs in a particular context in a previous post!

Some cryptographic algorithms are patented, yes. I'm not sure how
that's relevent. I do not know of any academic cryptographers that
regularly look through the U.S. patent system. Patents are not a
peer-reviewed publication. If an academic (in any discipline)
presented a list of patents to his tenure review board, they would not
be considered publication.

Patents are not relevent to academic publication. There are academics
who also patent. Ritter is an example of someone who does not publish
(in an academic sense) but does patent. His writings are generally
ignored by the academic community. I'm sorry this is true; I'd like
it to be different.

If I wrote a paper on Ritters designs, citing his patents and Usenet
postings and webpages, I believe that I would have a lot of trouble
getting it published.

>> >Let me perhaps quote what you wrote previously in this thread to
>> >illustrate the inconsistency of your logic, particularly in reference
>> >to Ritter's work. In response to Ritter's post of 20 Oct. 00:40:21
>> >you wrote on 26 Oct 03:59:50 :
>> >
>> >   I invite you to submit a paper, based on your patent #5,727,062
>> >   ("Variable Size Block Ciphers") to the 1999 Fast Software Encryption
>> >   workshop. I believe it will be published.
>> >
>> >However on 28 Oct 15:32:00 you wrote
>> >
>> >   There are Ritter's designs. Any of these algorithms could
>> >   potentially be cryptanalyzed by amateurs.
>> >
>> >Unless you are firmly of the opinion that FSE workshop is at such
>> >a (low) level such that it readily accepts papers presenting algorithms
>> >that could potentially be cryptanalyzed by amateurs (I am afraid the
>> >program committee of FES workshop would be angry with you) the
>> >above two paragraphs are entirely incompatible with each other in my
>> >humble view.
>>
>> Being on the program committee of FSE, I can categorically state that
>> the conference accepts papers preventing algorithms that could

Oops. That's "presenting" up there.

>> potentially be cryptanalyzed by amateurs. In FSE 97 we saw ICE and
>> TwoPrime, both of which were easy cryptanalyses. In SAC 96 we saw
>> Akelarre, which could have been cryptanalyzed by someone with not a
>> lot of skill. I assure you, the program committee is angry with
>> itself when a medeocre design slips in, but it happens. FSE is
>> supposed to publish cipher designs, so that there is fodder to be
>> analyzed.
>
>I find it is surprising that someone on the program committee can
>be so inconsiderate of the committee and make use of the name of the
>workshop to argue for his own issues in logically entirely inconsitent
>manner. Please say in clear terms if what I pointed out as
>contradiction is false.

Near as I can tell your "contradiction" is:

        I would like Ritter to submit his designs to FSE.

        I believe that Ritter's designs could potentially be
        cryptanalyzed by amateurs.

I see no contradiction there.

        1. Some designs published at FSE could be (or could have
        been) cryptanalyzed by amateurs. The committee is not
        perfect; never has been. Clunkers slip through. I gave some
        exxamples above.

        2. Ritters designs could potentiall be cryptanalyzed by
        amateurs. The "potentially" indicates that maybe I am
        wrong. However, Ritter's deisgns are designed to be
        scalable, so there are by definition "toy" versions that are
        much easier to analyze than large versions.

What do you see as the logical and entirely inconsistent manner of
this argument? Do you think that I am wrong in saying that some FSE
designs are medeocre? If so, please read the cryptanalyses of the
algorithms mentioned above. Do you believe that Ritter's designs are
a priori unanalyzable by amateurs? If so, please read Ritter's
postings on his toy versions.

I'm really trying to help here. I am not being inconsiderate to the program committee of FSE. I am not making use of the FSE name to argue my own position. I don't have a position. I have more than my share of ad hominum arguments on sci.crypt, and I would appreciate a little bit of curtesy.

Bruce

Bruce
****
Bruce Schneier, President, Counterpane Systems Phone: 612-823-1098
101 E Minnehaha Parkway, Minneapolis, MN 55419 Fax: 612-823-1590
Free crypto newsletter. See: http://www.counterpane.com

Subject: Re: Memo to the Amateur Cipher Designer
Date: Fri, 30 Oct 1998 19:10:13 +0100
From: Mok-Kong Shen <mok-kong.shen@stud.uni-muenchen.de>
Message-ID: <363A0105.81B31DB5@stud.uni-muenchen.de>
References: <3639d8bf.741333@news.visi.com>
Newsgroups: sci.crypt
Lines: 73

Bruce Schneier wrote:
>

> Some cryptographic algorithms are patented, yes. I'm not sure how
> that's relevent. I do not know of any academic cryptographers that
> regularly look through the U.S. patent system. Patents are not a
> peer-reviewed publication. If an academic (in any discipline)
> presented a list of patents to his tenure review board, they would not
> be considered publication.

This is not true. Applications for patents are examined by a number of professionals in the corresponding fields to ensure that the ideas are really novel and useful. There are huge data bases maintained by the patent offices and are carefully checked to ensure patents are not given to some one bring forth duplication or near duplication of prior art. You can certainly refer to a patent number (with title) as one of your (valid) literature reference. A good and hence accepted paper can on the other hand well be, say, an overview of certain currently interesting part of a discipline and contains nothing new. If one seldem sees references to patents, it is because they are quite few in number relative to the normal papers in any field.

>
> Patents are not relevent to academic publication. There are academics
> who also patent. Ritter is an example of someone who does not publish
> (in an academic sense) but does patent. His writings are generally
> ignored by the academic community. I'm sorry this is true; I'd like
> it to be different.

Patents cannot be ignored by the academic community. If one develops a new cipher, he needs to know whether he doesn't infringe on someone's patents. It is a legal issue that concerns him. Certainly one could ignore that like one could ignore possible violation of other laws. As far as I know R&D in organic chemistry and pharmacy quite often have to consider the patent problem.

>
> If I wrote a paper on Ritters designs, citing his patents and Usenet
> postings and webpages, I believe that I would have a lot of trouble
> getting it published.

> >> > I invite you to submit a paper, based on your patent #5,727,062
> >> > ("Variable Size Block Ciphers") to the 1999 Fast Software Encryption
> >> > workshop. I believe it will be published.

Isn't there some contradiction between these two paragraphs (given your opinions on patents above) ?

>
> Near as I can tell your "contradiction" is:
>
> I would like Ritter to submit his designs to FSE.
>
> I believe that Ritter's designs could potentially be
> cryptanalyzed by amateurs.
>
> I see no contradiction there.

As far as I know in scientific conferences there are always much much more submissions than can be taken up. Certainly there would be plenty of good papers from professionals that would be rejected by the coming FSE because of the capacity problem. As member of the program committee you really needn't fear that there would be too few good submissions to make the workshop successful. No propagada is needed at all. I don't yet see why you are particularly inclined to look for manuscripts coming from amateurs instead of from your fellow professionals or would-be professionals. Are Ritter's patents that interest you? But that seems to contradict what you said about patents.

M. K. Shen

Subject: Re: Memo to the Amateur Cipher Designer
Date: Fri, 30 Oct 1998 18:41:08 GMT
From: schneier@counterpane.com (Bruce Schneier)
Message-ID: <363b062f.2065537@news.visi.com>
References: <363A0105.81B31DB5@stud.uni-muenchen.de>
Newsgroups: sci.crypt
Lines: 98

On Fri, 30 Oct 1998 19:10:13 +0100, Mok-Kong Shen
<mok-kong.shen@stud.uni-muenchen.de> wrote:

>Bruce Schneier wrote:
>>
>
>> Some cryptographic algorithms are patented, yes. I'm not sure how
>> that's relevent. I do not know of any academic cryptographers that
>> regularly look through the U.S. patent system. Patents are not a
>> peer-reviewed publication. If an academic (in any discipline)
>> presented a list of patents to his tenure review board, they would not
>> be considered publication.
>
>This is not true. Applications for patents are examined by a number
>of professionals in the corresponding fields to ensure that the
>ideas are really novel and useful. There are huge data bases maintained
>by the patent offices and are carefully checked to ensure patents are
>not given to some one bring forth duplication or near duplication of
>prior art. You can certainly refer to a patent number (with title)
>as one of your (valid) literature reference. A good and hence accepted
>paper can on the other hand well be, say, an overview of certain
>currently interesting part of a discipline and contains nothing new.
>If one seldem sees references to patents, it is because they are quite
>few in number relative to the normal papers in any field.

Patents are not considered peer-reviewed publications in academia.
Please confirm this with academics in other disciplines. I do not
want you to take my word for this.

Patent applications are generally reviewed by one examiner, not "a
number of professionals in the corresponding fields." As far as I
know, the US patent office cannot, by law, use outside professionals.
Please confirm this with your patent attorney. Do not take my word
for it.

Patent applications are checked for patentability, not quality of
research. Again, please confirm this with your attorney.

One seldom sees references to patents because people who write papers
don't look through patent databases like they look through journals.
Again, please check this with other academics.

>As far as I know in scientific conferences there are always much much
>more submissions than can be taken up. Certainly there would be plenty
>of good papers from professionals that would be rejected by the coming
>FSE because of the capacity problem. As member of the program
>committee you really needn't fear that there would be too few good
>submissions to make the workshop successful.

There is usually many more submissions than accepted papers, and one
of the ways to judge a conference is the acceptance rate. Crypto and
Eurocrypt, for example, have acceptance rates around 25%. Some of the
lessar cryptography conferences have much higher acceptance rate--I
konw of one that was around 80%--and the quality of accepted papers
are much lower because of this. There was one crypto conference that
was cancelled because they did not receive enough quality papers.
Other conferences take medeocre papers. Other conferences take
average papers and fewer papers. CARDIS 98 is an example of this; I
was on the program committee and we just didn't get enough quality
papers.

Except for well-known and popular conferences, program committees
always fear not getting enough quality papers to make for a successful
conference. It's a big program in many disciplines, and a problem in
the lesser cryptography conferences. (Look at the Australian
conference, for example.)

>No propagada is needed
>at all. I don't yet see why you are particularly inclined to look for
>manuscripts coming from amateurs instead of from your fellow
>professionals or would-be professionals.

All the committees I know of review all papers received, regardless of
where they are from. Crypto and Eurocrypt have blind refereeing,
which means that the referees don't know whether the papers come from
"fellow professionals or would-be professionals." I have pulled three
proceedings off my shelves at pseudo-random, and all contain some
papers not from people in the "mainstream."

>Are Ritter's patents
>that interest you? But that seems to contradict what you said

>about patents.

I'm not sure what I said about patents that indicates that Ritter's would not interest me. Ritter's work interests me, and his patents are one of the few places I can read about it. I admit that I have not studied his designs in any detail, but that is more due to lack of time (and the lessened possibility of publishable results) than lack of interest.

Are we getting somewhere?

Bruce
****
Bruce Schneier, President, Counterpane Systems Phone: 612-823-1098
101 E Minnehaha Parkway, Minneapolis, MN 55419 Fax: 612-823-1590
Free crypto newsletter. See: http://www.counterpane.com

Subject: Re: Memo to the Amateur Cipher Designer
Date: Mon, 02 Nov 1998 17:34:09 +0100
From: Mok-Kong Shen <mok-kong.shen@stud.uni-muenchen.de>
Message-ID: <363DDF00.CD0B2609@stud.uni-muenchen.de>
References: <363b062f.2065537@news.visi.com>
Newsgroups: sci.crypt
Lines: 305

Bruce Schneier wrote:
>
> On Fri, 30 Oct 1998 19:10:13 +0100, Mok-Kong Shen
> <mok-kong.shen@stud.uni-muenchen.de> wrote:
>
> >Bruce Schneier wrote:
> >>
> >
> >> Some cryptographic algorithms are patented, yes. I'm not sure how
> >> that's relevent. I do not know of any academic cryptographers that
> >> regularly look through the U.S. patent system. Patents are not a
> >> peer-reviewed publication. If an academic (in any discipline)
> >> presented a list of patents to his tenure review board, they would not
> >> be considered publication.
> >
> >This is not true. Applications for patents are examined by a number
> >of professionals in the corresponding fields to ensure that the
> >ideas are really novel and useful. There are huge data bases maintained
> >by the patent offices and are carefully checked to ensure patents are
> >not given to some one bring forth duplication or near duplication of
> >prior art. You can certainly refer to a patent number (with title)
> >as one of your (valid) literature reference. A good and hence accepted
> >paper can on the other hand well be, say, an overview of certain
> >currently interesting part of a discipline and contains nothing new.
> >If one seldem sees references to patents, it is because they are quite
> >few in number relative to the normal papers in any field.
>
> Patents are not considered peer-reviewed publications in academia.
> Please confirm this with academics in other disciplines. I do not
> want you to take my word for this.

Lest our discussions go eventually into wrong directions because there are essential differences between your and my definitions, I like very much first to obtain agreement with you on the definition of the term 'peer review', which is an important concept in our current debate. I don't know a dictionary giving exactly that. But in Webster Third New International Dictionary there is the following for the word 'peer':

1 a: one that is of the seme or equal standing (as in law, rank, quality, age, ability) with another : EQUAL.

b: a fellow citizen.

2: archaic: COMPANION, FELLOW.

So I think it is appropriate to define 'peer review' in a scientific discipline to be review by knowledgeable, capable persons in that field. Since we have been employing the dichotomy amateur and professional, I suggest that we can agree to understand 'peer review' to be review by professionals. That is, the people doing peer review are from the general scientific community and NOT limited to those from academia (those having positions in the universities and in particular those with tenure). If you are not in (at least near) agreement with this, then it is unconditionally necessary that we carry out a discussion on this essential terminology problem, otherwise our writings to and fro would be senseless.

In the following, I assume that there is no terminology difficulty in the said point. The immediately following paragraphs of my comment will also cover materials that concern later parts that are to be responded by me (this lumping together is done in order to achieve some better continuity of my argumentation.)

With the said assumption, your point above could be rephrased into : 'Patents are not considered publications reviewed by the professionals in the respective field.' Is there difficulty here for you to accept this? If yes, please state your counter-

argument in a bit detail and give a proposal of your rephrasing so that we can further discuss on that.

Before waiting your follow-up I temporary assume that you agree on that and carry on my argumentation based on that. Now as I said, a patent office employs professionals in diverse scientific fields with the purpose to ensure that the goals of the patent laws are achieved, i.e., to say roughly, that patents are only issued to inventions that are really novel and potentially useful. It may well be argued whether these employed professionals are good or bad, but I think this is the responsibility of the government. To make an analogon, we can't for purpose of general discussions ask whether the judges in courts are capable or incapable. We have to assume that the government has done a proper job and has choosen capable people to take the positions of judges. (If many people are of the opinion that the government has done a bad job, they should do something in politics, including revolution.) In our case it is my point that an examination at the patent office (by the professionals there) has the equivalent (scientific) quality as the normally designated 'peer reviews' that are done by persons chosen by the journals. (It is my personal opinion that the quality might probably be higher, since the examiners would obtain difficulties if they were found to be careless in scrutinizing the patent applications while a reviewer of a normal journal paper doesn't have material consequences if his evaluation turns out to be wrong. But let's ignore that in order not to overly lenghten our discussions.) Do you agree? If not, please elaborate your view.

Now each professional, however knowledgeable he is, has only a finite limited knowledge. An application may happen to be at a higer knowledge level than his current stand. I don't know exactly what happens in that case in the different countries. But I'll like to make a plausible guess. Suppose someone applies for a crypto patent involving very high mathematical knowledge of the kind of Prof. Schnorr's patent. I can very well imagine that the professionals at the patent offices might feel uncertain of their own judgement and would seek outside advice, i.e. from other government institutions (the so-called 'mutual assistance' of government institutions, in this case quite likely to involve a three-lettered agency), from universities or from industry that are known to have the expertise. It is in any case the responsiblity of the examiner(s) (lastly the resposibility of the patent office as a legal person) to see to it that their job is properly done (i.e. is done as is desired by the patent law). I don't see that we should question in the framework of this thread whether that responsibility is in fact fulfilled or not. See also the previous paragraph. (One would also not question whether a university has chosen professors that really do good teaching jobs in similar contexts of discussion.)

Now I like to argue for the degree of acknowlegement of the value (status) of patents by the professionals (the scientific community) with special reference to cryptology. In A. J. Menezes et al. a whole chapter, Chap. 15, is devoted to 'Patents and Standards'. There they write:

> This chapter discusses two topics which have significant impact on the use of cryptology in practice: patents and standards. At their best, cryptographic patents make details of significant new processes and efficient techiques publicly available, thereby increasing awareness and promoting use; at their worst, they limit or stifle the use of such techniques due to licencing requirement.

This in my view clearly shows that patents plays in important role in R&D in cryptology. Like them or not, they ARE there and cannot be willingly ignored if one conducts proper business. In Menezes's book the number of pages actually dealing with patents amounts to about 2% of the total pages of the book, a certainly non-trivial figure. Patent publications cannot be ignored simply because they are government publications and thus in some respect different from the normal journals. (I guess someone doing a serious implementation would not rely entirely on informations obtainable from some text book on cryptology but consult the original document MBS FIPS PUB 45-1.)

>
> Patent applications are generally reviewed by one examiner, not "a
> number of professionals in the corresponding fields." As far as I
> know, the US patent office cannot, by law, use outside professionals.
> Please confirm this with your patent attorney. Do not take my word
> for it.

Two points. One: If the responsible examiner feels his competence is well good enough in a specific case, I believe he will assume his responsibility of deciding alone. But there is nothing wrong in that. There are plenty of courts precided by a single judge. Does that mean injustice? But patents are NOT granted simply because an examiner or a group of examiner employed by the patent office think it is o.k. A draft of a patent must be published and within a certain time period anybody can bring forth objections to the pending patent. I suppose you know that large firms like IBM have fairly sizable patent divisions that not only work closely with the research divisions to ensure that patentable research results indeed get patented but also to constantly watch whether a patent applied by somebody else infringes on existing or pending patents of their own.) So this gives in fact possibility of examination by a much much wider

circle of professionals than is the case with a paper that is submitted to a journal (with a few referees only). Sometimes journals publish papers containing results duplicating prior publications (leading to letters to the editor), which is a consequence of the fact that the manuscripts are only available to a rather small number of referees before the papers actually come out.
Two: Your challenge 'Please confirm this with your patent attorney.' is an inappropriate type of challenge. If I were to say to you 'In Germany (or even US), as far as I know the law in the field X is such and such. Please confirm this with your lawyer.', would you accept that challenge? (Note that to consult a lawyer costs money, not only time.) If you know something for sure and like to use it for your argument in the debate, it is YOUR job to bring forth the evidence, not mine!

>
> Patent applications are checked for patentability, not quality of
> research. Again, please confirm this with your attorney.

Again two points: One: What do you exactly define as 'qualtity of research'? And as 'patentability'? Two: Analogous response to the last part of my previous paragraph.

>
> One seldom sees references to patents because people who write papers
> don't look through patent databases like they look through journals.
> Again, please check this with other academics.

In your own book, Applied Cryptography, 2nd ed., a quick (hence maybe incomplete) scan gives the following references (in number) to patent publications:

111	223	323	326	327	328	331	388
514	554	667	678	684	685	686	710
722	1013	1066	1072	1086	1087	1330	1389
1483	1489	1614					

You give in your book the exact number, title and date of issue of each patent and even take care to give references to patent numbers of different countries in case the same thing is granted a patent by different countries. Why do you take so much trouble if you are convinced that patent publications are unimportant for research and learning and hence barely of value to readers of your book? The above references make 2 percent of the total references of your book. Do you think that this is too low a figure? Note that patents are not easy to obtain. Their applications and maintenance cost money, in some cases quite a lot. Some people just give up their right to patent because of finance or do not apply patents for reasons of principle (e.g. not to stifle the technical developments in cryptology). By nature patent publications cannot be as abundant as normal scientific papers. (Analogon: one can't expect to encounter as many of the families of the nobles as the common people, simply because there are fewer of the former in existence.) Otherwise lots of scientists would be earning money from patent licences.

>
> >As far as I know in scientific conferences there are always much much
> >more submissions than can be taken up. Certainly there would be plenty
> >of good papers from professionals that would be rejected by the coming
> >FSE because of the capacity problem. As member of the program
> >committee you really needn't fear that there would be too few good
> >submissions to make the workshop successful.
>
> There is usually many more submissions than accepted papers, and one
> of the ways to judge a conference is the acceptance rate. Crypto and
> Eurocrypt, for example, have acceptance rates around 25%. Some of the
> lessar cryptography conferences have much higher acceptance rate--I
> konw of one that was around 80%--and the quality of accepted papers
> are much lower because of this. There was one crypto conference that
> was cancelled because they did not receive enough quality papers.
> Other conferences take medeocre papers. Other conferences take
> average papers and fewer papers. CARDIS 98 is an example of this; I
> was on the program committee and we just didn't get enough quality
> papers.

I suppose an interesting and relevant figure in the present context is that of FSE 98, which has an acceptance rate of 50%. But that certainly imply that there were sufficiently ample good candidates than could be accepted by the workshop. I happened to have some 'internal' knowledge of some scientific conferences (not cryptology). If thses cases could be generalized, than the (high) majority of submissions are good candidates, i.e. bad papers are seldom submitted. (I guess that the distinguished names of the members of the program committe alone usually suffice to 'frighten' away submitters of poor papers.)

>
> Except for well-known and popular conferences, program committees
> always fear not getting enough quality papers to make for a successful
> conference. It's a big program in many disciplines, and a problem in
> the lesser cryptography conferences. (Look at the Australian
> conference, for example.)

See above. It seems reasonalbe to 'extrapolate' (anticipate) for FSE 99 using the data of FSE 98.

>
> >No propagada is needed
> >at all. I don't yet see why you are particularly inclined to look for
> >manuscripts coming from amateurs instead of from your fellow
> >professionals or would-be professionals.
>
> All the committees I know of review all papers received, regardless of
> where they are from. Crypto and Eurocrypt have blind refereeing,
> which means that the referees don't know whether the papers come from
> "fellow professionals or would-be professionals." I have pulled three
> proceedings off my shelves at pseudo-random, and all contain some
> papers not from people in the "mainstream."

>
> >Are Ritter's patents
> >that interest you? But that seems to contradict what you said
> >about patents.
>
> I'm not sure what I said about patents that indicates that Ritter's
> would not interest me. Ritter's work interests me, and his patents
> are one of the few places I can read about it. I admit that I have
> not studied his designs in any detail, but that is more due to lack of
> time (and the lessened possibility of publishable results) than lack
> of interest.

> Are we getting somewhere?

You snipped a part from the previous post which I take the liberty
to reproduce below. I like very much to have your answers to the
question therein before commenting (effectively) to what you wrote
above. Here is the reproduction:

> >
> > If I wrote a paper on Ritters designs, citing his patents and Usenet
> > postings and webpages, I believe that I would have a lot of trouble
> > getting it published.
>
> > >> >   I invite you to submit a paper, based on your patent #5,727,062
> > >> >   ("Variable Size Block Ciphers") to the 1999 Fast Software Encryption
> > >> >   workshop. I believe it will be published.
>
> Isn't there some contradiction between these two paragraphs (given
> your opinions on patents above) ?

M. K. Shen

Subject: Re: Memo to the Amateur Cipher Designer
Date: 2 Nov 1998 12:01:40 -0500
From: juola@mathcs.duq.edu (Patrick Juola)
Message-ID: <71kohk$r3j$1@quine.mathcs.duq.edu>
References: <363DDF00.CD0B2609@stud.uni-muenchen.de>
Newsgroups: sci.crypt
Lines: 126

In article <363DDF00.CD0B2609@stud.uni-muenchen.de>,
Mok-Kong Shen <mok-kong.shen@stud.uni-muenchen.de> wrote:
>Bruce Schneier wrote:
>> On Fri, 30 Oct 1998 19:10:13 +0100, Mok-Kong Shen
>> <mok-kong.shen@stud.uni-muenchen.de> wrote:
>> >Bruce Schneier wrote:
>> >>
>> >
>> >> Some cryptographic algorithms are patented, yes. I'm not sure how
>> >> that's relevent. I do not know of any academic cryptographers that
>> >> regularly look through the U.S. patent system. Patents are not a
>> >> peer-reviewed publication. If an academic (in any discipline)
>> >> presented a list of patents to his tenure review board, they would not
>> >> be considered publication.
>> >
>> >This is not true. Applications for patents are examined by a number
>> >of professionals in the corresponding fields to ensure that the
>> >ideas are really novel and useful....
>>
>> Patents are not considered peer-reviewed publications in academia....
>
>Lest our discussions go eventually into wrong directions because
>there are essential differences between your and my definitions, I
>like very much first to obtain agreement with you on the definition
>of the term 'peer review', which is an important concept in our
>current debate. I don't know a dictionary giving exactly that. But
>in Webster Third New International Dictionary there is the following
>for the word 'peer':
>
>   1 a: one that is of the seme or equal standing (as in law, rank,
>   quality, age, ability) with another : EQUAL.
>
>   b: a fellow citizen.
>
>   2: archaic: COMPANION, FELLOW.
>
>So I think it is appropriate to define 'peer review' in a scientific

>discipline to be review by knowledgeable, capable persons in that
>field.

Um, Mr. Shen, the definition of what is regarded as "peer review" is not for you (nor Mr. Schneier) to make; the people whose opinions matter are generally those sitting on tenure and promotion boards, funding and review committees for scientific foundations, and editorial boards of journals and conferences. Irrespective of what you -- or Webster's Third -- might wish to say about the definition of the word "peer," the phrase "peer review" has a very specific meaning to those people. You might as well complain because the "National Football League" has a definition of "football" that differs from other sports called "football." It's a largely irrelevant quibble of interest to very few and that will change no one's opinion.

Similarly, "peer review" does not simply mean "review by one's peers" -- I can't just walk a paper around my department, have everyone in the department review it and initial it, and then claim that that document is "peer-reviewed."

> Since we have been employing the dichotomy amateur and
>professional, I suggest that we can agree to understand 'peer review'
>to be review by professionals. That is, the people doing peer review
>are from the general scientific community and NOT limited to those
>from academia (those having positions in the universities and in
>particular those with tenure).

The "general scientific community" is *NOT* restricted only to academics (or only to those with tenure, &c.) For example, most of the active researchers at Bell Labs, Xerox PARC, or IBM T.J. Watson would all be regarded as members of the scientific community, as "peers" for a review board. You can confirm this fairly easily by looking at the program committee for any major meeting (or the editorial board of any journal.

On the other hand, patent agents are *NOT*, in general, members of the scientific community. They generally don't know enough about science.

Despite your claim,

> Now as I said, a
>patent office employs professionals in diverse scientific fields
>with the purpose to ensure that the goals of the patent laws are
>achieved, i.e., to say roughly, that patents are only issued to
>inventions that are really novel and potentially useful. It may well
>be argued whether these employed professionals are good or bad, but
>I think this is the responsibility of the government. To make an
>analogon, we can't for purpose of general discussions ask whether the
>judges in courts are capable or incapable. We have to assume that
>the government has done a proper job and has choosen capable people
>to take the positions of judges. (If many people are of the opinion
>that the government has done a bad job, they should do something
>in politics, including revolution.)

... this is, in fact, exactly what's happened. The overall standing of patent review is sufficiently low that the people who have the authority to decide what does and doesn't constitute "peer review" have decided that patents don't cut it.

> In our case it is my point that
>an examination at the patent office (by the professionals there)
>has the equivalent (scientific) quality as the normally designated
>'peer reviews' that are done by persons chosen by the journals.

This point is simply speaking untrue. And even if it were true, it's neither your place nor Mr. Schneier's to correct it, as you don't sit on the relevant committees.

>> Patent applications are generally reviewed by one examiner, not "a
>> number of professionals in the corresponding fields." As far as I
>> know, the US patent office cannot, by law, use outside professionals.
>> Please confirm this with your patent attorney. Do not take my word
>> for it.
>
>Two points. One: If the responsible examiner feels his competence is
>well good enough in a specific case, I believe he will assume his
>responsibility of deciding alone. But there is nothing wrong in that.
>There are plenty of courts precided by a single judge.

Irrelevant comparison. Judges don't issue "peer reviewed" decisions, nor are their decisions regarded as "scientifically" valid. And, in fact, the law is very careful in how it treats issues of "scientific questions" because of the fact that, *BY PRESUMPTION*, judges are not scientific experts.

-kitten

Subject: Re: Memo to the Amateur Cipher Designer
Date: 2 Nov 1998 22:41:42 GMT
From: jpeschel@aol.com (JPeschel)
Message-ID: <19981102174142.20334.00002760@ng141.aol.com>

References: <363DDF00.CD0B2609@stud.uni-muenchen.de>
Newsgroups: sci.crypt
Lines: 17

>Mok-Kong Shen <mok-kong.shen@stud.uni-muenchen.de>writes a lot of
stuff about patents that I snipped.

Why not go here and look around:

Http://www.uspto.gov/web/offices/pac/doc/general/index.html

Joe

Joe Peschel
D.O.E. SysWorks
http://members.aol.com/jpeschel/index.htm

Subject: Re: Memo to the Amateur Cipher Designer
Date: Tue, 03 Nov 1998 19:01:17 +0100
From: Mok-Kong Shen <mok-kong.shen@stud.uni-muenchen.de>
Message-ID: <363F44ED.FEF5E3C@stud.uni-muenchen.de>
References: <363b062f.2065537@news.visi.com>
Newsgroups: sci.crypt
Lines: 532

Note: For motivation given at the end of this post, I select the
alternative to reply in the group. For this reason, the incoming
material is nowhere snipped. Apology for its length.

Bruce Schneier wrote:
>
> Look, this isn't worth this long an email. I will read briefly, respond
> briefly, and then probably drop the matter.
>
> At 05:34 PM 11/2/98 +0100, Mok-Kong Shen wrote:
> >Bruce Schneier wrote:
> >>
> >> On Fri, 30 Oct 1998 19:10:13 +0100, Mok-Kong Shen
> >> <mok-kong.shen@stud.uni-muenchen.de> wrote:
> >>
> >> >Bruce Schneier wrote:
> >> >>
> >> >
> >> >> Some cryptographic algorithms are patented, yes. I'm not sure how
> >> >> that's relevent. I do not know of any academic cryptographers that
> >> >> regularly look through the U.S. patent system. Patents are not a
> >> >> peer-reviewed publication. If an academic (in any discipline)
> >> >> presented a list of patents to his tenure review board, they would not
> >> >> be considered publication.
> >> >
> >> >This is not true. Applications for patents are examined by a number
> >> >of professionals in the corresponding fields to ensure that the
> >> >ideas are really novel and useful. There are huge data bases maintained
> >> >by the patent offices and are carefully checked to ensure patents are
> >> >not given to some one bring forth duplication or near duplication of
> >> >prior art. You can certainly refer to a patent number (with title)
> >> >as one of your (valid) literature reference. A good and hence accepted
> >> >paper can on the other hand well be, say, an overview of certain
> >> >currently interesting part of a discipline and contains nothing new.
> >> >If one seldem sees references to patents, it is because they are quite
> >> >few in number relative to the normal papers in any field.
> >>
> >> Patents are not considered peer-reviewed publications in academia.
> >> Please confirm this with academics in other disciplines. I do not
> >> want you to take my word for this.
> >
> >Lest our discussions go eventually into wrong directions because
> >there are essential differences between your and my definitions, I
> >like very much first to obtain agreement with you on the definition
> >of the term 'peer review', which is an important concept in our
> >current debate. I don't know a dictionary giving exactly that. But
> >in Webster Third New International Dictionary there is the following
> >for the word 'peer':
> >
> >   1 a: one that is of the seme or equal standing (as in law, rank,
> >   quality, age, ability) with another : EQUAL.
> >
> >   b: a fellow citizen.
> >
> >   2: archaic: COMPANION, FELLOW.
>
> I define peer review as appearing in a conference proceedings, workshop
> proceedings, or journal that is peer reviewed. I mean the term in the
> academic sense, as a professor might use in "peer reviewed publication."
>
> I am not defending "shoulds." I am stating reality. Please confirm with
> others.

>
> >So I think it is appropriate to define 'peer review' in a scientific
> >discipline to be review by knowledgeable, capable persons in that
> >field. Since we have been employing the dichotomy amateur and
> >professional,
>
> ...which I don't like at all...
>
> >I suggest that we can agree to understand 'peer review'
> >to be review by professionals. That is, the people doing peer review
> >are from the general scientific community and NOT limited to those
> >from academia (those having positions in the universities and in
> >particular those with tenure).
>
> Ah, academics only use academic peer-review for things like tenure.\
> This may be our disagreement.
>
> >If you are not in (at least near)
> >agreement with this, then it is unconditionally necessary that we
> >carry out a discussion on this essential terminology problem,
> >otherwise our writings to and fro would be senseless.
>
> We can just choose to drop the matter, which seems much easier.

A few points for reflexion: Are the journals and proceedings published to be read exclusively by the academics (and not by the scientific community as a whole)? It is the value (and hence meaning) of 'peer review' for the scientific community as a whole and not the possible deviating meaning of someones in the universities that really (practically) counts for the material that the publishing bodies provide to the scientific community as services (for which they pay their money). There are scientific conferences where the majority of the program committe, even including the chair, are not from universities. Are those who are not academics not 'peers' doing the review and are not equivalent to those who have university positions in the process? Why is it necessary to lay weight at all on being academics or not academics in the present context? Is a good scientist, say one from a research division of an industrial firm, necessarily less qualified for the review work? Why couldn't we forget in the present discussion the special professional subclass academics and talk instead only in terms of the professionals in the scientific community (the superclass) as I suggested?

>
> >In the following, I assume that there is no terminology difficulty
> >in the said point. The immediately following paragraphs of my
> >comment will also cover materials that concern later parts that are
> >to be responded by me (this lumping together is done in order to
> >achieve some better continuity of my argumentation.)
> >
> >With the said assumption, your point above could be rephrased into :
> >'Patents are not considered publications reviewed by the
> >professionals in the respective field.' Is there difficulty here
> >for you to accept this? If yes, please state your counter-
> >argument in a bit detail and give a proposal of your rephrasing so
> >that we can further discuss on that.
>
> This is true.
>
> >Before waiting your follow-up I temporary assume that you agree on
> >that and carry on my argumentation based on that. Now as I said, a
> >patent office employs professionals in diverse scientific fields
> >with the purpose to ensure that the goals of the patent laws are
> >achieved, i.e., to say roughly, that patents are only issued to
> >inventions that are really novel and potentially useful. It may well
> >be argued whether these employed professionals are good or bad, but
> >I think this is the responsibility of the government. To make an
> >analogon, we can't for purpose of general discussions ask whether the
> >judges in courts are capable or incapable. We have to assume that
> >the government has done a proper job and has choosen capable people
> >to take the positions of judges. (If many people are of the opinion
> >that the government has done a bad job, they should do something
> >in politics, including revolution.) In our case it is my point that
> >an examination at the patent office (by the professionals there)
> >has the equivalent (scientific) quality as the normally designated
> >'peer reviews' that are done by persons chosen by the journals. (It
> >is my personal opinion that the quality might probably be higher,
> >since the examiners would obtain difficulties if they were found
> >to be careless in scrutinizing the patent applications while a
> >reviewer of a normal journal paper doesn't have material
> >consequences if his evaluation turns out to be wrong. But let's
> >ignore that in order not to overly lenghten our discussions.) Do
> >you agree? If not, please elaborate your view.
>
> Have you ever met patent examiners? I have. They are the ones
> who can't get real jobs in the field. Patent review does not equal
> per review in an academic sense.

Yes, I in fact have met some, though longtime ago and not in discussions about any specific patents but about building data base facilities for searching patents. For the 'equality' issue see previous post and below.

>
> >Now each professional, however knowledgeable he is, has only a

> >finite limited knowledge. An application may happen to be at a higer
> >knowledge level than his current stand. I don't know exactly what
> >happens in that case in the different countries. But I'll like to
> >make a plausible guess. Suppose someone applies for a crypto patent
> >involving very high mathematical knowledge of the kind of Prof.
> >Schnorr's patent. I can very well imagine that the professionals at
> >the patent offices might feel uncertain of their own judgement and
> >would seek outside advice, i.e. from other government institutions
> >(the so-called 'mutual assistance' of government institutions, in
> >this case quite likely to involve a three-lettered agency), from
> >universities or from industry that are known to have the expertise.
> >It is in any case the responsiblity of the examiner(s) (lastly the
> >resposibility of the patent office as a legal person) to see to it
> >that their job is properly done (i.e. is done as is desired by the
> >patent law). I don't see that we should question in the framework
> >of this thread whether that responsibility is in fact fulfilled or
> >not. See also the previous paragraph. (One would also not question
> >whether a university has chosen professors that really do good
> >teaching jobs in similar contexts of discussion.)
>
> The patent office does not have the same aims as an academic reviewer.
> The patent office doesn't care if a partular crypto algorithm is any good,
> only if it is unique. Remember that.

Try to arbitrarily assemble some pieces of metal in the fashion of
some (very) modern artists into a 'machine' and apply for a patent
(claiming it to be a new crypto hardware). That 'machine' is certainly
unique (an unique art object in the whole world!). According to what
you wrote, one can get a patent. Can this be true?? (The 'machine' can
be so constructed so as to allow an 'interpretation' of transforming
a bit 1 to 0 and vice versa, thus substantiating its 'claim' of
'encrypting' informations and hence being a crypto hardware.)

>
> >Now I like to argue for the degree of acknowlegement of the value
> >(status) of patents by the professionals (the scientific community)
> >with special reference to cryptology. In A. J. Menezes et al. a
> >whole chapter, Chap. 15, is devoted to 'Patents and Standards'.
> >There they write:
> >
> >   This chapter discusses two topics which have significant impact
> >   on the use of cryptology in practice: patents and standards. At
> >   their best, cryptographic patents make details of significant
> >   new processes and efficient techiques publicly available,
> >   thereby increasing awareness and promoting use; at their worst,
> >   they limit or stifle the use of such techniques due to licencing
> >   requirement.
> >
> >This in my view clearly shows that patents plays in important role
> >in R&D in cryptology. Like them or not, they ARE there and cannot
> >be willingly ignored if one conducts proper business. In Menezes's
> >book the number of pages actually dealing with patents amounts to
> >about 2% of the total pages of the book, a certainly non-trivial
> >figure. Patent publications cannot be ignored simply because they
> >are government publications and thus in some respect different from
> >the normal journals. (I guess someone doing a serious implementation
> >would not rely entirely on informations obtainable from some text
> >book on cryptology but consult the original document MBS FIPS PUB 45-1.)
>
> Okay. Fine.
> >
> >> Patent applications are generally reviewed by one examiner, not "a
> >> number of professionals in the corresponding fields." As far as I
> >> know, the US patent office cannot, by law, use outside professionals.
> >> Please confirm this with your patent attorney. Do not take my word
> >> for it.
> >
> >Two points. One: If the responsible examiner feels his competence is
> >well good enough in a specific case, I believe he will assume his
> >responsibility of deciding alone. But there is nothing wrong in that.
> >There are plenty of courts precided by a single judge. Does that
> >mean injustice? But patents are NOT granted simply because an
> >examiner or a group of examiner employed by the patent office think
> >it is o.k. A draft of a patent must be published and within a certain
> >time period anybody can bring forth objections to the pending patent.
> >I suppose you know that large firms like IBM have fairly sizable
> >patent divisions that not only work closely with the research
> >divisions to ensure that patentable research results indeed get
> >patented but also to constantly watch whether a patent applied by
> >somebody else infringes on existing or pending patents of their own.)
> >So this gives in fact possibility of examination by a much much wider
> >circle of professionals than is the case with a paper that is
> >submitted to a journal (with a few referees only). Sometimes journals
> >publish papers containing results duplicating prior publications
> >(leading to letters to the editor), which is a consequence of the
> >fact that the manuscripts are only available to a rather small number
> >of referees before the papers actually come out.
>
> Have you ever submitted a patent, been involved in patent prosecution,
> or fought a patent in court? I wish reality worked like the above
> paragraph.

I haven't submitted patents. But it is certainly permitted that I
discuss about the matter? In the real world there are also judges
who are incapable and decide wrongly. As I argued previously patents

are NOT issued simply because the examiners employed by the patent officices think they are o.k. There is a public review period, in which the pending patents are invariably critically examined by professionals of firms whose own patents potentially could be infringed on. (Truly critically, because big revenues may under circumstances be involved.) Tell me how a pending crypto patent under the watching eyes of professionals of those firms like IBM can have a easier chance of getting passed (i.e. not objected to in case of infringement) than a paper subjected to a journal (in case of duplication of prior results). (Recently in a mailing list someone said he could not exploit the idea of using faces in a scheme for passphrase entry because IBM has a patent that is very broad to cover that.) In which sense is such a public review less effective (stringent) than a 'peer review' in a journal? If despite this fact some academics have a different opinion, then we of the scientific community CAN really ignore that opinion. (Is there any scientist willing to adhere to doctrines (upheld maybe by some 'authorities') that are evidently wrong?)

>
> >Two: Your challenge 'Please confirm this with your patent attorney.'
> >is an inappropriate type of challenge. If I were to say to you 'In
> >Germany (or even US), as far as I know the law in the field X is
> >such and such. Please confirm this with your lawyer.', would you
> >accept that challenge? (Note that to consult a lawyer costs money,
> >not only time.) If you know something for sure and like to use it
> >for your argument in the debate, it is YOUR job to bring forth the
> >evidence, not mine!
>
> THen I won't be doing my job, because this discussion isn't worth
> that much time. Sorry; I don't mean to be rude. I read sci.crypt for
> fun, not to find more work to do.

I don't see you are responding to my point here at all. I claimed that the quoted challenge is inappropriate. What has that to do with fun or not fun in sci.crypt or work or not work?? You challenged other people to consult lawers. That is not only WORK but monetary expenses!!

> >
> >> Patent applications are checked for patentability, not quality of
> >> research. Again, please confirm this with your attorney.
> >
> >Again two points: One: What do you exactly define as 'qualtity of
> >research'? And as 'patentability'? Two: Analogous response to the
> >last part of my previous paragraph.
>
> "Quality of research" is whether the algoritm of secure or not.
> Patentability is VERY broadly defined. Again, talk to your attorney.

I refer you to my previous point on inappropriateness of the said challenge. You think that the predicate 'secure' or 'not secure' necessarily has to come from academics?? We discussed sometime ago in the group that the very definition of the strength of encryption algorithms is a difficult (highly debatable) issue. You simply claimed that the one concept of the two above is broader than the other, giving no clues at all of what you have exactly in mind. Please give some appropriate and clear definitions so that others can see your point and be able to discuss.

>
> >> One seldom sees references to patents because people who write papers
> >> don't look through patent databases like they look through journals.
> >> Again, please check this with other academics.
> >
> >In your own book, Applied Cryptography, 2nd ed., a quick (hence
> >maybe incomplete) scan gives the following references (in number)
> >to patent publications:
> >
> > 111 223 323 326 327 328 331 388
> > 514 554 667 678 684 685 686 710
> > 722 1013 1066 1072 1086 1087 1330 1389
> > 1483 1489 1614
>
> Seldom does not equal never. I purposely referenced patents. Open a
> CRYPTO proceedings and to the same measurement.

How 'seldom' is the stuff here really? Patents issued are inventions that have already been achieved by some scientists. Normally a patent has a certain coverage, preventing some more or less variants being done by others. (I remember recently in a mailing list there was some disscussion of that matter in relation to Schnorr's patent.) A crypto paper is most likely to cite a patent if there is some significant analysis of that patented crypto, which is seldom if we assume that the patented cryptos are mostly fairly strong and even then it would mostly suffice to use the common name of the crypto without pedantically giving the patent number, date of issue, etc. in the list of references of the paper. This suffices to explain the phenomenon you described.

BTW, when you designed and submitted your AES candidate, did you have no concerns at all about patent issues? I simply can't imagine that.

>

> >You give in your book the exact number, title and date of issue of
> >each patent and even take care to give references to patent numbers
> >of different countries in case the same thing is granted a patent by
> >different countries. Why do you take so much trouble if you are
> >convinced that patent publications are unimportant for research
> >and learning and hence barely of value to readers of your book? The
> >above references make 2 percent of the total references of your book.
> >Do you think that this is too low a figure? Note that patents are not
> >easy to obtain. Their applications and maintenance cost money,
> >in some cases quite a lot. Some people just give up their right to
> >patent because of finance or do not apply patents for reasons of
> >principle (e.g. not to stifle the technical developments in
> >cryptology). By nature patent publications cannot be as abundant as
> >normal scientific papers. (Analogon: one can't expect to encounter
> >as many of the families of the nobles as the common people, simply
> >because there are fewer of the former in existence.) Otherwise lots
> >of scientists would be earning money from patent licences.
> >
> >> >As far as I know in scientific conferences there are always much much
> >> >more submissions than can be taken up. Certainly there would be plenty
> >> >of good papers from professionals that would be rejected by the coming
> >> >FSE because of the capacity problem. As member of the program
> >> >committee you really needn't fear that there would be too few good
> >> >submissions to make the workshop successful.
> >>
> >> There is usually many more submissions than accepted papers, and one
> >> of the ways to judge a conference is the acceptance rate. Crypto and
> >> Eurocrypt, for example, have acceptance rates around 25%. Some of the
> >> lessar cryptography conferences have much higher acceptance rate--I
> >> konw of one that was around 80%--and the quality of accepted papers
> >> are much lower because of this. There was one crypto conference that
> >> was cancelled because they did not receive enough quality papers.
> >> Other conferences take medeocre papers. Other conferences take
> >> average papers and fewer papers. CARDIS 98 is an example of this; I
> >> was on the program committee and we just didn't get enough quality
> >> papers.
> >
> >I suppose an interesting and relevant figure in the present context
> >is that of FSE 98, which has an acceptance rate of 50%. But that
> >certainly imply that there were sufficiently ample good candidates
> >than could be accepted by the workshop. I happened to have some
> >'internal' knowledge of some scientific conferences (not cryptology).
> >If thses cases could be generalized, than the (high) majority of
> >submissions are good candidates, i.e. bad papers are seldom
> >submitted. (I guess that the distinguished names of the members of
> >the program committe alone usually suffice to 'frighten' away
> >submitters of poor papers.)
>
> Crypto is different. There are MANY poor papers. I am on the
> EUCROCRYPT '99 committee, and there are about 1/4 papers that
> can be imediately rejected.

>
> >> Except for well-known and popular conferences, program committees
> >> always fear not getting enough quality papers to make for a successful
> >> conference. It's a big program in many disciplines, and a problem in
> >> the lesser cryptography conferences. (Look at the Australian
> >> conference, for example.)
> >
> >See above. It seems reasonalbe to 'extrapolate' (anticipate) for
> >FSE 99 using the data of FSE 98.
>
> Extrapolate away. You are free not to believe me, although I'm not sure
> why you think I would lie.

Is there any word of mine suspecting that you lie in this point? I
said only that for FSE 99 you don't need to fear for not having
insufficient good submissions in view of the statistics of FSE 98.

BTW in scientific discussions a false statement expressed with
all sincerity remains false and a true statement from a liar
(if the statement happens to be true because he misunderstood
the matter) is true. So the question of to lie or not to lie
plays no role in scientific discussions. It is only the truth-
finding that is at issue.

>
> >> >No propagada is needed
> >> >at all. I don't yet see why you are particularly inclined to look for
> >> >manuscripts coming from amateurs instead of from your fellow
> >> >professionals or would-be professionals.
> >>
> >> All the committees I know of review all papers received, regardless of
> >> where they are from. Crypto and Eurocrypt have blind refereeing,
> >> which means that the referees don't know whether the papers come from
> >> "fellow professionals or would-be professionals." I have pulled three
> >> proceedings off my shelves at pseudo-random, and all contain some
> >> papers not from people in the "mainstream."
> >
> >>
> >> >Are Ritter's patents
> >> >that interest you? But that seems to contradict what you said
> >> >about patents.
> >>

> >> I'm not sure what I said about patents that indicates that Ritter's
> >> would not interest me. Ritter's work interests me, and his patents
> >> are one of the few places I can read about it. I admit that I have
> >> not studied his designs in any detail, but that is more due to lack of
> >> time (and the lessened possibility of publishable results) than lack
> >> of interest.
> >
> >> Are we getting somewhere?
> >
> >You snipped a part from the previous post which I take the liberty
> >to reproduce below. I like very much to have your answers to the
> >question therein before commenting (effectively) to what you wrote
> >above. Here is the reproduction:
> >
> >> >
> >> > If I wrote a paper on Ritters designs, citing his patents and Usenet
> >> > postings and webpages, I believe that I would have a lot of trouble
> >> > getting it published.
> >>
> >> > >> >   I invite you to submit a paper, based on your patent #5,727,062
> >> > >> >   ("Variable Size Block Ciphers") to the 1999 Fast Software
> Encryption
> >> > >> >   workshop. I believe it will be published.
> >>
> >> Isn't there some contradiction between these two paragraphs (given
> >> your opinions on patents above) ?
> >
> No. There is no contradiction. If you see one, I am sorry. I may respond
> a bit
> in public if you continue the thread, but I don't see the point in continuing.

O.K. I'll say what I think in this connection. You are a professional, Ritter is an amateur. You have published a lot in the established journals and have written a (the) bestseller in cryptology. It can be safely said that to write a paper on one and the same subject you have much much more advantages over Ritter at the current moment of time because of your ample experience (in writing papers) and the extent of your knowledge in the field. (Ritter could certainly catch up, but that would at least take some time.) If such a paper written by you would have a lot of trouble of getting published, how can you 'believe' that a paper written by Ritter will be published? You can know to some degree (self-feeling) how good or bad the paper would be if you indeed attempt that. How do you know the quality of Ritter's paper which you invite him to submit? Independent of the quality of the scientific content, what happens if he presents the stuff very poorly? How can you 'believe' that the submission will be successful?

In the Memo it is your golden rule that amateurs and would-be professionals should first do solid analysis work and publish some appreciable analysis results before publishing designs. This is a highly valuable advice acknowledged by many in this thread. But aren't you presently recommending Ritter to break your own rule?

There are other aspects not entirely to be neglected. You have now already a definite 'belief' of the quality of Ritter's future work without seeing his manuscript ('it will be published'). Would this 'belief' somehow unconciously affect your evaluation of Ritter's paper when you come to actually referee it? (Compare the issue of prejudice with repect to judges in courts.) Even if the editorial procedure is such that Ritter's name is not on the manuscript, the probability is almost 1 that you can identify Ritter's paper and since Ritter is going to cite his patent number other members of the program commitee can do the same with high probability. Now you have given already some evaluation of his (yet non-existant) paper in that you 'believe' it will be published and give this openly in sci.crpyt. Would other members of the program committe somehow unconciously be affectd in their evaluation of the same by your (premature) opinion? In other words, would the 'neutralness' and 'independence' of the program commitee become thereby a litle bit not 100% perfect? With the Memo you have probably done something fairly positive to the amatuers of sci.crypt (I expressed my appreciation in another post), but in my humble view you have done something negative to FSE 99 and perphaps even modifies a little bit the view of the common people (excluding those in the academia) towards the 'peer reviewed' publications. This is in my personal view rather unfortunate.

All in all I have come through the stuffs discussed to the personal opinion that Terry Ritter is going to have a very thorny road before him, if he indeed takes your recommendation to write a paper on his design for submission to FSE 99. I like very much to warn him of that. For that purpose I have to argue with you, attempting to show the logical inconsitency of your argumentation and discussing with you openly in the group so that any mistakes and errors on my part may be readily discovered by other people. (This is why I have chosen to post this to the group.)

M. K. Shen

Subject: Re: Memo to the Amateur Cipher Designer
Date: Fri, 06 Nov 1998 07:49:39 GMT
From: schneier@counterpane.com (Bruce Schneier)
Message-ID: <3648a66c.1950739@news.visi.com>

References: <363F44ED.FEF5E3C@stud.uni-muenchen.de>
Newsgroups: sci.crypt
Lines: 200

On Tue, 03 Nov 1998 19:01:17 +0100, Mok-Kong Shen
<mok-kong.shen@stud.uni-muenchen.de> wrote:
>Try to arbitrarily assemble some pieces of metal in the fashion of
>some (very) modern artists into a 'machine' and apply for a patent
>(claiming it to be a new crypto hardware). That 'machine' is certainly
>unique (an unique art object in the whole world!). According to what
>you wrote, one can get a patent. Can this be true?? (The 'machine' can
>be so constructed so as to allow an 'interpretation' of transforming
>a bit 1 to 0 and vice versa, thus substantiating its 'claim' of
>'encrypting' informations and hence being a crypto hardware.)

I believe you would be amazed by what gets through the patent office.
The only thing they regularly catch are perpetual motion machines; bad
cryptography will fly right over their heads. (For heavens sake, they
can't even week out impossible compression patents.)

>I haven't submitted patents. But it is certainly permitted that I
>discuss about the matter? In the real world there are also judges
>who are incapable and decide wrongly. As I argued previously patents
>are NOT issued simply because the examiners employed by the patent
>officices think they are o.k.

Yes. You argued that previously. You are wrong. Patents are issued
because the patent examiner who has the application has allowed some
of the claims.

>There is a public review period, in
>which the pending patents are invariably critically examined by
>professionals of firms whose own patents potentially could be
>infringed on.

I don't know what country you live in, but in the U.S. there is no
public review period for pending patents. Patents are made public
after they are granted, and not before. (Note: this is changing now
that we have signed up to GATT. U.S. patents will be made public 18
months after filing, regardless of award status.)

>(Truly critically, because big revenues may under
>circumstances be involved.) Tell me how a pending crypto patent
>under the watching eyes of professionals of those firms like IBM
>can have a easier chance of getting passed (i.e. not objected to in
>case of infringement) than a paper subjected to a journal (in case of
>duplication of prior results).

Sure. IBM does not see the patent application.

>(Recently in a mailing list someone
>said he could not exploit the idea of using faces in a scheme for
>passphrase entry because IBM has a patent that is very broad to
>cover that.) In which sense is such a public review less effective
>(stringent) than a 'peer review' in a journal?

It's not public. There is no such public review. In very competitive
industries--pharmasuticals come to mind--companies watch foreign
filings for clies as to what the competition is doing. But I know of
no instance of a company trying to block a patent from being awarded.
If you have such examples, please let me know.

>> >Two: Your challenge 'Please confirm this with your patent attorney.'
>> >is an inappropriate type of challenge. If I were to say to you 'In
>> >Germany (or even US), as far as I know the law in the field X is
>> >such and such. Please confirm this with your lawyer.', would you
>> >accept that challenge? (Note that to consult a lawyer costs money,
>> >not only time.) If you know something for sure and like to use it
>> >for your argument in the debate, it is YOUR job to bring forth the
>> >evidence, not mine!
>>
>> THen I won't be doing my job, because this discussion isn't worth
>> that much time. Sorry; I don't mean to be rude. I read sci.crypt for
>> fun, not to find more work to do.
>
>I don't see you are responding to my point here at all. I claimed that
>the quoted challenge is inappropriate. What has that to do with
>fun or not fun in sci.crypt or work or not work?? You challenged other
>people to consult lawers. That is not only WORK but monetary expenses!!

Look, I didn't challenge anyone to consult a lawyer. I suggested that
if you don't believe me, you should consider asking someone else who
may know. I really don't care enough about this argument to spend the
time necessary to convince you.

And there are some excellent books on patent law by Nolo Press.

>BTW, when you designed and submitted your AES candidate, did you
>have no concerns at all about patent issues? I simply can't imagine
>that.

Of course I had concern. But if I was simply writing an academic
paper, I wouldn't.

>Is there any word of mine suspecting that you lie in this point? I

>said only that for FSE 99 you don't need to fear for not having
>insufficient good submissions in view of the statistics of FSE 98.

Agreed. Because there is so much good research, and the AES process is bringing even more of it out of the woodwork, I expect an excellent program for FSE 98.

>O.K. I'll say what I think in this connection. You are a professional,
>Ritter is an amateur. You have published a lot in the established
>journals and have written a (the) bestseller in cryptology. It can
>be safely said that to write a paper on one and the same subject you
>have much much more advantages over Ritter at the current moment of
>time because of your ample experience (in writing papers) and the
>extent of your knowledge in the field. (Ritter could certainly catch
>up, but that would at least take some time.) If such a paper written
>by you would have a lot of trouble of getting published, how can you
>'believe' that a paper written by Ritter will be published? You can
>know to some degree (self-feeling) how good or bad the paper would
>be if you indeed attempt that. How do you know the quality of
>Ritter's paper which you invite him to submit? Independent of the
>quality of the scientific content, what happens if he presents the
>stuff very poorly? How can you 'believe' that the submission will be
>successful?

Thank you. I finally understand what your issue is. You believe that for me to 1) say that I would have trouble getting a paper published on Ritter's stuff published, and 2) suggest that Ritter write one himself, is contradictory.

I wish you said that in the beginning; it would have saved a lot of bandwidth.

A few things:

1. Ritter's paper would be a design paper, not an analysis paper. Design papers appear at FSE. I think this is a good thing. I believe it would be easier for Ritter to get a paper published at FSE with some of his design ideas than for someone to get a paper published at FSE analyzing some of his design ideas (unless they were published first).

2. I was specifically suggesting that Ritter publish in FSE, in hopes that he would join the committee.

3. You know, you're right.

>In the Memo it is your golden rule that amateurs and would-be
>professionals should first do solid analysis work and publish
>some appreciable analysis results before publishing designs. This is
>a highly valuable advice acknowledged by many in this thread. But
>aren't you presently recommending Ritter to break your own rule?

Indeed. I am.

>There are other aspects not entirely to be neglected. You have
>now already a definite 'belief' of the quality of Ritter's future
>work without seeing his manuscript ('it will be published'). Would
>this 'belief' somehow unconciously affect your evaluation of Ritter's
>paper when you come to actually referee it?

Oh, definitely. But if it wasn't any good, the committee would not let it in.

>(Compare the issue of
>prejudice with repect to judges in courts.) Even if the editorial
>procedure is such that Ritter's name is not on the manuscript, the
>probability is almost 1 that you can identify Ritter's paper and
>since Ritter is going to cite his patent number other members of the
>program commitee can do the same with high probability. Now you
>have given already some evaluation of his (yet non-existant) paper
>in that you 'believe' it will be published and give this openly
>in sci.crpyt. Would other members of the program committe somehow
>unconciously be affectd in their evaluation of the same by your
>(premature) opinion?

No. Trust me on this one.

>In other words, would the 'neutralness' and
>'independence' of the program commitee become thereby a litle bit not
>100% perfect? With the Memo you have probably done something fairly
>positive to the amatuers of sci.crypt (I expressed my appreciation
>in another post), but in my humble view you have done something
>negative to FSE 99 and perphaps even modifies a little bit the view
>of the common people (excluding those in the academia) towards the
>'peer reviewed' publications. This is in my personal view rather
>unfortunate.

Oh, I get it. Honestly, I don't think it will be a problem. The paper would have been judged on its own merits. I believe that his design writings, as I have seen them, are the sorts of things that FSE accepts. That's all.

>All in all I have come through the stuffs discussed to the personal
>opinion that Terry Ritter is going to have a very thorny road before
>him, if he indeed takes your recommendation to write a paper on his

>design for submission to FSE 99. I like very much to warn him of that.
>For that purpose I have to argue with you, attempting to show the
>logical inconsitency of your argumentation and discussing with you
>openly in the group so that any mistakes and errors on my part may
>be readily discovered by other people. (This is why I have chosen to
>post this to the group.)

Got it. I hope we're done now.

Bruce
****
Bruce Schneier, President, Counterpane Systems Phone: 612-823-1098
101 E Minnehaha Parkway, Minneapolis, MN 55419 Fax: 612-823-1590
Free crypto newsletter. See: http://www.counterpane.com

Subject: Re: Memo to the Amateur Cipher Designer
Date: Fri, 06 Nov 1998 16:05:19 GMT
From: schneier@counterpane.com (Bruce Schneier)
Message-ID: <36441e09.660949@news.visi.com>
References: <3648a66c.1950739@news.visi.com>
Newsgroups: sci.crypt
Lines: 13

On Fri, 06 Nov 1998 07:49:39 GMT, schneier@counterpane.com (Bruce
Schneier) wrote:

>2. I was specifically suggesting that Ritter publish in FSE, in hopes
>that he would join the committee.

Oops. I meant "community."

Bruce
****
Bruce Schneier, President, Counterpane Systems Phone: 612-823-1098
101 E Minnehaha Parkway, Minneapolis, MN 55419 Fax: 612-823-1590
Free crypto newsletter. See: http://www.counterpane.com

Subject: Re: Memo to the Amateur Cipher Designer
Date: Tue, 10 Nov 1998 10:54:36 +0100
From: Mok-Kong Shen <mok-kong.shen@stud.uni-muenchen.de>
Message-ID: <36480D5C.7676EE5D@stud.uni-muenchen.de>
References: <3648a66c.1950739@news.visi.com>
Newsgroups: sci.crypt
Lines: 32

Bruce Schneier wrote:

>
> I don't know what country you live in, but in the U.S. there is no
> public review period for pending patents. Patents are made public
> after they are granted, and not before. (Note: this is changing now
> that we have signed up to GATT. U.S. patents will be made public 18
> months after filing, regardless of award status.)

I have been able to verify that there are public reviews for German,
British and European patents. I have no easy access to US laws.
It is strange but I can well comprehend the radical difference of
patenting in US from patenting in other countries, since there
apprears to me to be differences in the underlying general 'philosophy'
of laws as is manisfested in the issue of carrying of guns by common
people which is prohibited in almost all nations.

> Thank you. I finally understand what your issue is. You believe that
> for me to 1) say that I would have trouble getting a paper published
> on Ritter's stuff published, and 2) suggest that Ritter write one
> himself, is contradictory.
>
> I wish you said that in the beginning; it would have saved a lot of
> bandwidth.

My sincere apology all to readers of the group for having wasted
bandwidth. (But I thought it would have been sufficient to simply
ask (twice) the question of the existence of contradiction in point
without appending too many words.) Sorry.

M. K. Shen

Subject: Re: Memo to the Amateur Cipher Designer
Date: Wed, 11 Nov 1998 09:08:54 +0100
From: Mok-Kong Shen <mok-kong.shen@stud.uni-muenchen.de>
Message-ID: <36494616.1500C031@stud.uni-muenchen.de>
References: <36480D5C.7676EE5D@stud.uni-muenchen.de>
Newsgroups: sci.crypt
Lines: 50

Mok-Kong Shen wrote:

> My sincere apology all to readers of the group for having wasted
> bandwidth. (But I thought it would have been sufficient to simply

> ask (twice) the question of the existence of contradiction in point
> without appending too many words.) Sorry.

Someone pointed out to me that I could have saved more bandwidth if I had paid attention to the following fact:

> In the academically 'peer reviewed' paper by J. Kelsey, B. Schneier, D. Wagner, C. Hall, Cryptanalytic Attacks on Pseudo-random Number Generators (in S. Vaudenay (ed), Fast Software Encryption, Springer, 1998), there is the following entry in the references:
>
> [Koc95] P. Kocher, post to sci.crypt Internet newsgroup (message-ID pckDIr4Ar.L4zQnetcom.com), 4 Dec 1995

This clearly refutes the claim that references to newsgroup articles are unworthy of good scientific papers and shows up the existence of some more logical inconsistencies than I had been able to uncover.

It is noteworthy that in the same paper there are also references to Web page URL and to ftp URL. So these are also valid references for good scientific papers. Should patent publications be less worthy of being cited scientifically?? (In scientific publications one sometimes even see such references as 'Privte communiction from XXX'. I personally consider these to be rather useless since the reader has practically no possibility to access these communications.)

I am taking this opportunity to say a bit more about peer review of patent publications. A scientific paper is reviewed by a number of peers before publishing. Afterwards, there is public review. A reader may object by writing a 'letter to editor'. For a patent (in the case of US, not other countries!) there is no review by peers before patent publication. But afterwards the document is examined by the professionals in the patent divisions of a number of firms whose own patents may be concerned. If there are objections there will be a legal issue in court. (This is at least true in countries other than US.) So what is the essential difference in the value of information transmission (to the scientific community which comprises not solely of the academics but many more) between a scientific paper and a patent document? If the academics choose to ignore the patent publications and claim that only the papers in journals edited by them are scientific contributions (I doubt this), then they are not practicizing science but 'religion'! (Note though, that in ancient times science was not separated from religion.)

M. K. Shen

Subject: Re: Memo to the Amateur Cipher Designer
Date: 11 Nov 1998 08:50:05 GMT
From: sax@rmovt.rply.ce.chalmers.se (Stefan Axelsson)
Message-ID: <72bj3tbbk1@nyheter.chalmers.se>
References: <36494616.1500C031@stud.uni-muenchen.de>
Newsgroups: sci.crypt
Lines: 34

In article <36494616.1500C031@stud.uni-muenchen.de>,
Mok-Kong Shen <mok-kong.shen@stud.uni-muenchen.de> wrote:

>   [Koc95] P. Kocher, post to sci.crypt Internet newsgroup (message-
>   ID pckDIr4Ar.L4zQnetcom.com), 4 Dec 1995
>
>This clearly refutes the claim that references to newsgroup articles
>are unworthy of good scientific papers and shows up the existence
>of some more logical inconsistencies than I had been able to uncover.
>
>It is noteworthy that in the same paper there are also references to
>Web page URL and to ftp URL. So these are also valid references
>for good scientific papers.

No, in general, they, currently, are, not. The problem of what to do with URL:s is a somewhat debated topic in academic circles today, and no real consensus has been reached. However, the majority of researchers recognise that there are difficult, and fundamental problems with referring to URL:s, or other forms of transient communication.

The requirements for a "good" scientific paper, whatever that may be, today, I would say, would exclude even references to obscure publications, where they not crucial to the treatment of the subject.

A newsgroup posting, I would say, is (almost) right up there with "personal conversation with N.N.", dejanews or not. It detracts from the scientific value of the paper, it doesn't add to it!

Stefan,
--
Stefan Axelsson Chalmers University of Technology
sax@rmovt.rply.ce.chalmers.se Dept. of Computer Engineering
(Remove "rmovt.rply" to send mail.)

Subject: Re: Memo to the Amateur Cipher Designer
Date: Wed, 11 Nov 1998 16:42:23 -0600

From: jgfunj@EnqvbSerrGrknf.pbz (W T Shaw)
Message-ID: <jgfunj-1111981642240001@dialup164.itexas.net>
References: <72bj3tbbk1@nyheter.chalmers.se>
Newsgroups: sci.crypt
Lines: 15

In article <72bj3tbbk1@nyheter.chalmers.se>,
sax@rmovt.rply.ce.chalmers.se (Stefan Axelsson) wrote:
>
> A newsgroup posting, I would say, is (almost) right up there with
> "personal conversation with N.N.", dejanews or not. It detracts from
> the scientific value of the paper, it doesn't add to it!
>
Scientific truth is what is valuable, preferable to that of a paper
published through a process that might ignore aspects by limiting debate
of the particulars.
--

The public is harder to steamroller than some might think.

Decrypt with ROT13 to get correct email address.

Subject: Re: Memo to the Amateur Cipher Designer
Date: 12 Nov 1998 13:09:15 GMT
From: sax@rmovt.rply.ce.chalmers.se (Stefan Axelsson)
Message-ID: <72emlrjsa1@nyheter.chalmers.se>
References: <jgfunj-1111981642240001@dialup164.itexas.net>
Newsgroups: sci.crypt
Lines: 35

In article <jgfunj-1111981642240001@dialup164.itexas.net>,
W T Shaw <jgfunj@EnqvbSerrGrknf.pbz> wrote:
>Scientific truth is what is valuable, preferable to that of a paper
>published through a process that might ignore aspects by limiting debate
>of the particulars.

Not being a native speaker, I had difficulty parsing that, but I'm
going to take one last stab anyway, feel free to have the last word.

"Scientific truth" is the operative phrase here. While the fact
that the author of a scientific paper has chosen to refer to sources
that the reader cannot himself verify, not in, and of, itself detracts
from the intrinsic "truth" of the paper's stated position, it does
detract heavily from the "scientific" part of your statement. The
process is specifically *not* designed to "limit the discussion of the
particulars" but, instead to further such discussion!

Now, of course there are references, and references, but if one
resorts to to building ones argument on a reference that the reader
cannot himself verify, then of course one must question why, and if,
that reference is to be included at all, it is of little, or no, value
to the reader. *)

The peer review process may not be perfect, not many human endeavors
are, but it's the best there is.

*) No reference to the paper by Bruce Schneier intended. I haven't
   read the paper in question, and thus could not possibly comment, on
   the kind of reference intended.

Stefan,
--
Stefan Axelsson Chalmers University of Technology
sax@rmovt.rply.ce.chalmers.se Dept. of Computer Engineering
(Remove "rmovt.rply" to send mail.)

Subject: Re: Memo to the Amateur Cipher Designer
Date: Thu, 12 Nov 1998 14:19:41 GMT
From: "Joseph K. Nilaad" <jknilaad@xoommail.com>
Message-ID: <364AEE7D.FCF@xoommail.com>
References: <72bj3tbbk1@nyheter.chalmers.se>
Newsgroups: sci.crypt
Lines: 31

Stefan Axelsson wrote:
>
[snip]
> No, in general, they, currently, are, not. The problem of what to do
> with URL:s is a somewhat debated topic in academic circles today, and
> no real consensus has been reached. However, the majority of
> researchers recognise that there are difficult, and fundamental
> problems with referring to URL:s, or other forms of transient
> communication.
Those researchers you've mentioned, should stop using the damn computers
and hand writing manuscript with the manual type writers and send it to
their peers via regular mail. I can see that URL may be short life, but
as long as it lives, it should be considered valid reference.

>
> The requirements for a "good" scientific paper, whatever that may be,
> today, I would say, would exclude even references to obscure
> publications, where they not crucial to the treatment of the subject.

>
> A newsgroup posting, I would say, is (almost) right up there with
> "personal conversation with N.N.", dejanews or not. It detracts from
> the scientific value of the paper, it doesn't add to it!
>
Maybe you should stop using news group. But how contradiction that is
since you do read dejanews.

Joseph K. Nilaad
Nature is simple and beautiful...
Life is too short to appreciate it all...

Subject: Re: Memo to the Amateur Cipher Designer
Date: 12 Nov 1998 10:19:38 -0500
From: juola@mathcs.duq.edu (Patrick Juola)
Message-ID: <72euaadob1@quine.mathcs.duq.edu>
References: <364AEE7D.FCF@xoommail.com>
Newsgroups: sci.crypt
Lines: 46

In article <364AEE7D.FCF@xoommail.com>,
Joseph K. Nilaad <jknilaad@xoommail.com> wrote:
>Stefan Axelsson wrote:
>>
>[snip]
>> No, in general, they, currently, are, not. The problem of what to do
>> with URL:s is a somewhat debated topic in academic circles today, and
>> no real consensus has been reached. However, the majority of
>> researchers recognise that there are difficult, and fundamental
>> problems with referring to URL:s, or other forms of transient
>> communication.
>Those researchers you've mentioned, should stop using the damn computers
>and hand writing manuscript with the manual type writers and send it to
>their peers via regular mail. I can see that URL may be short life, but
>as long as it lives, it should be considered valid reference.

Yes, but what happens when it no longer lives?

The point of references isn't to impress people with how widely
you've read. It's to provide explanations for when someone else can't
follow your work or needs to look at the foundations.

For example, "It has been observed (Flintstone and Rubble, 1999) that
20% of microfleems are subradiant." Observed under what conditions?
The microfleems in my lab are running at damn nearly 40% subradiant.
Does this mean that I'm using an odd back of microfleems, or has the
population changed over time, or that I'm measuring them wrongly?
So I go to the journals and figure out just what definition F&R are
using for subradiantness.

(Flintstone, p.c.) is marginally helpful. I can at least phone Fred
up and ask him what the hell he was doing. *If* I know who Flintstone
is and what lab he's at now and if he still remembers doing the work.

But http://www.bedrock.com/~fleems isn't nearly as helpful if the
domain no longer exists and I can't even tell who did the work to
phone him.

Sending everything to my peers through regular mail would make the
problem worse, not better. I don't have access to the private letters
that Flintstone writes you. What I need is some way to confirm the
Flintstone-Rubble data, and the only way I can do that is if you
give me a location that you and I both *know* I can find it.

        -kitten

Subject: Re: Memo to the Amateur Cipher Designer
Date: Wed, 18 Nov 1998 03:24:44 GMT
From: "Joseph K. Nilaad" <jknilaad@xoommail.com>
Message-ID: <36523DFC.7E11@xoommail.com>
References: <72euaadob1@quine.mathcs.duq.edu>
Newsgroups: sci.crypt
Lines: 47

Patrick Juola wrote:
>
[snip]
> >as long as it lives, it should be considered valid reference.
>
> Yes, but what happens when it no longer lives?
Likewise, what if the publishers like Random house no longer exist. So
what If referenced URL no longer exist. At least you're being *honest*
about it. There are so many good things come out of URL, that's
undeniable.

> The point of references isn't to impress people with how widely
> you've read. It's to provide explanations for when someone else can't
> follow your work or needs to look at the foundations.
Agreed. But lack of them, isn't impressive either.

[snip]
> But http://www.bedrock.com/~fleems isn't nearly as helpful if the
> domain no longer exists and I can't even tell who did the work to
> phone him.
I neither agree nore disagree here. It's tough issue. However, what if
the URL still exists.

Now let say, you see something that Fred talks his buddy about how to
improve the ride for Flint mobile from http://www.bedrock.com/~fleems.
The improvement has a lot of text and demonstration graphics such that
you can't absorb information in a reasonable amount of time or maybe you
want to study further. You like the idea and want to use it sometime in
the future. What will you do? Most likely, you will print the page(s)
for reading later. Most browsers now print the URL address and date.

One day your peer, Steve, drops by. You tell Steve about the
improvement idea. He likes it and want to know more. What will you
tell him? Are you afraid to tell him that it's from
http://www.bedrock.com/~fleems or it's your because URL doesn't count.

> that Flintstone writes you. What I need is some way to confirm the
> Flintstone-Rubble data, and the only way I can do that is if you
> give me a location that you and I both *know* I can find it.
I've read from September 1998 issue that a company may deliver 800MHz
x86 CPU in Q1 1999.

Joseph K. Nilaad
Nature is simple and beautiful...
Life is too short to appreciate it all...

Subject: Re: Memo to the Amateur Cipher Designer
Date: 18 Nov 1998 08:45:35 -0500
From: juola@mathcs.duq.edu (Patrick Juola)
Message-ID: <72uj1voqq1@quine.mathcs.duq.edu>
References: <36523DFC.7E11@xoommail.com>
Newsgroups: sci.crypt
Lines: 37

In article <36523DFC.7E11@xoommail.com>,
Joseph K. Nilaad <jknilaad@xoommail.com> wrote:
>Patrick Juola wrote:
>> The point of references isn't to impress people with how widely
>> you've read. It's to provide explanations for when someone else can't
>> follow your work or needs to look at the foundations.
>Agreed. But lack of them, isn't impressive either.
>
>[snip]
>> But http://www.bedrock.com/~fleems isn't nearly as helpful if the
>> domain no longer exists and I can't even tell who did the work to
>> phone him.
>I neither agree nore disagree here. It's tough issue. However, what if
>the URL still exists.
>
>Now let say, you see something that Fred talks his buddy about how to
>improve the ride for Flint mobile from http://www.bedrock.com/~fleems.
>The improvement has a lot of text and demonstration graphics such that
>you can't absorb information in a reasonable amount of time or maybe you
>want to study further. You like the idea and want to use it sometime in
>the future. What will you do? Most likely, you will print the page(s)
>for reading later. Most browsers now print the URL address and date.
>
>One day your peer, Steve, drops by. You tell Steve about the
>improvement idea. He likes it and want to know more. What will you
>tell him? Are you afraid to tell him that it's from
>http://www.bedrock.com/~fleems or it's your because URL doesn't count.

I'm likely to photocopy the page(s) for him (or loan them to him).
Particularly if I have any reason to suspect that the page has been
changed and/or gone 404.

The basic problem is that telling him that it's at
http:[...]/~fleems is borderline useless.

-kitten

Subject: Re: Memo to the Amateur Cipher Designer
Date: 23 Nov 1998 00:44:36 GMT
From: clvisser@gene.wins.uva.nl (Coen L.S. Visser)
Message-ID: <73ab5k$p8d$1@nbox.wins.uva.nl>
References: <72uj1voqq1@quine.mathcs.duq.edu>
Newsgroups: sci.crypt
Lines: 33

Joseph K. Nilaad <jknilaad@xoommail.com> wrote:
>Patrick Juola wrote:
>> The point of references isn't to impress people with how widely
>> you've read. It's to provide explanations for when someone else can't
>> follow your work or needs to look at the foundations.

> Agreed. But lack of them, isn't impressive either.

>> But http://www.bedrock.com/~fleems isn't nearly as helpful if the
>> domain no longer exists and I can't even tell who did the work to
>> phone him.

>I neither agree nore disagree here. It's tough issue. However, what if
>the URL still exists.

The problem is more serious than just a disappearing URL. What if the URL still exists, but the content has changed. That might give some semantical problems.

A writer is responsible for his or her references. If you want to use a webpage as reference mirror the specific page and make sure that the exact content that is refered to is available even if the original page changes or disappears. Example:

Patrick Reijnen (november 1998), Linux Hardware Compatibility HOWTO,
http://sunsite.unc.edu/LDP/HOWTO/Hardware-HOWTO.html also at
http://my.own.site/MyBookRefs/Hardware-HOWTO.html

There could be some copyright issues. These might be resolved by denying access to the mirror until the original disappears.

Regards,

        Coen Visser

Subject: Re: Memo to the Amateur Cipher Designer
Date: Mon, 23 Nov 1998 09:44:33 +0100
From: galactus@stack.nl (Arnoud "Galactus" Engelfriet)
Message-ID: <xBSW24uYOdQY089yn@stack.nl>
References: <73ab5k$p8d$1@nbox.wins.uva.nl>
Newsgroups: sci.crypt
Lines: 22

In article <73ab5k$p8d$1@nbox.wins.uva.nl>,
clvisser@gene.wins.uva.nl (Coen L.S. Visser) wrote:
> A writer is responsible for his or her references. If you want to use a
> webpage as reference mirror the specific page and make sure that the exact
> content that is refered to is available even if the original page changes
> or disappears.

How about downloading the relevant documents from the Web and putting
them on a CD-ROM, which is distributed together with the report? This
is what I'll be doing for my graduation project. The copyright issues
still exist, of course, and it may not be very practical for articles
that are published for a wide audience.

Greetings,

Arnoud

--
\/ Arnoud "Galactus" Engelfriet - galactus@stack.nl This space
    5th year Business & Computing Science student left blank
    URL: http://www.stack.nl/~galactus/ PGP: 0x416A1A35 intentionally.

Subject: Re: Memo to the Amateur Cipher Designer
Date: 23 Nov 1998 23:20:51 GMT
From: clvisser@gene.wins.uva.nl (Coen L.S. Visser)
Message-ID: <73cqkj$lb0$1@nbox.wins.uva.nl>
References: <xBSW24uYOdQY089yn@stack.nl>
Newsgroups: sci.crypt
Lines: 21

galactus@stack.nl (Arnoud "Galactus" Engelfriet) writes:
>In article <73ab5k$p8d$1@nbox.wins.uva.nl>,
>clvisser@gene.wins.uva.nl (Coen L.S. Visser) wrote:
>> A writer is responsible for his or her references. If you want to use a
>> webpage as reference mirror the specific page and make sure that the exact
>> content that is refered to is available even if the original page changes
>> or disappears.

>How about downloading the relevant documents from the Web and putting
>them on a CD-ROM, which is distributed together with the report? This
>is what I'll be doing for my graduation project. The copyright issues
>still exist, of course, and it may not be very practical for articles
>that are published for a wide audience.

That would be really nice of course but as you already state it has a
lot of practical problems.

Regards,

        Coen

Subject: Re: Memo to the Amateur Cipher Designer

Date: 23 Nov 1998 10:13:38 GMT
From: sax@rmovt.rply.ce.chalmers.se (Stefan Axelsson)
Message-ID: <73bcginis1@nyheter.chalmers.se>
References: <73ab5k$p8d$1@nbox.wins.uva.nl>
Newsgroups: sci.crypt
Lines: 34

In article <73ab5k$p8d$1@nbox.wins.uva.nl>,
Coen L.S. Visser <clvisser@gene.wins.uva.nl> wrote:

>The problem is more serious than just a disappearing URL. What if the
>URL still exists, but the content has changed. That might give some
>semantical problems. A writer is responsible for his or her
>references. If you want to use a webpage as reference mirror the
>specific page and make sure that the exact content that is refered to
>is available even if the original page changes or disappears.
>Example: Patrick Reijnen (november 1998), Linux Hardware
>Compatibility HOWTO,
>http://sunsite.unc.edu/LDP/HOWTO/Hardware-HOWTO.html also at
>http://my.own.site/MyBookRefs/Hardware-HOWTO.html There could be some
>copyright issues. These might be resolved by denying access to the
>mirror until the original disappears. Regards, Coen Visser

Only that doesn't work either, I've found many cases of papers, not
too old, where the author *himself* refers to his own papers, that
would supposedly be available through the web. Only, he was a PhD
student when he wrote them, and as you know, such aren't necessarily
around for all that long, after which most of their web pages fall
into that great big bit-bucket in the sky...

What is needed, is some other, resilient, long lasting, redundant
third party storage of references, such as a library is for printed
material today. I know I can count the British Library several orders
of magnitude more than I do the web... (But then they charge for the
service...)

Stefan,
--
Stefan Axelsson Chalmers University of Technology
sax@rmovt.rply.ce.chalmers.se Dept. of Computer Engineering
(Remove "rmovt.rply" to send mail.)

Subject: Re: Memo to the Amateur Cipher Designer
Date: 23 Nov 1998 14:28:27 GMT
From: clvisser@gene.wins.uva.nl (Coen L.S. Visser)
Message-ID: <73breb$cj4$1@nbox.wins.uva.nl>
References: <73bcginis1@nyheter.chalmers.se>
Newsgroups: sci.crypt
Lines: 28

sax@rmovt.rply.ce.chalmers.se (Stefan Axelsson) writes:
>Coen L.S. Visser <clvisser@gene.wins.uva.nl> wrote:

>>If you want to use a webpage as reference mirror the
>>specific page and make sure that the exact content that is refered to
>>is available even if the original page changes or disappears.

>Only that doesn't work either, I've found many cases of papers, not
>too old, where the author *himself* refers to his own papers, that
>would supposedly be available through the web. Only, he was a PhD
>student when he wrote them, and as you know, such aren't necessarily
>around for all that long, after which most of their web pages fall
>into that great big bit-bucket in the sky...

>What is needed, is some other, resilient, long lasting, redundant
>third party storage of references, such as a library is for printed
>material today. I know I can count the British Library several orders
>of magnitude more than I do the web... (But then they charge for the
>service...)

Yes, I agree, a third party storage of references would be the best thing.
That would take the burden of archiving from the shoulders of authors.
Libraries would be ideal for that task, they have the experience.
Research institutes and publishers also have the knowledge to do it.

Regards,

Coen Visser

Subject: Re: Memo to the Amateur Cipher Designer
Date: 14 Nov 1998 15:32:53 GMT
From: sax@rmovt.rply.ce.chalmers.se (Stefan Axelsson)
Message-ID: <72k7r5mju1@nyheter.chalmers.se>
References: <364AEE7D.FCF@xoommail.com>
Newsgroups: sci.crypt
Lines: 44

In article <364AEE7D.FCF@xoommail.com>,
Joseph K. Nilaad <jknilaad@xoommail.com> shared his lack of insight
with us, and wrote:

>Those researchers you've mentioned, should stop using the damn computers

>and hand writing manuscript with the manual type writers and send it to
>their peers via regular mail. I can see that URL may be short life, but
>as long as it lives, it should be considered valid reference.
[...]
>Maybe you should stop using news group. But how contradiction that is
>since you do read dejanews.

Look, the average time from research to publication in a refereed journal today is two years. Many/most of those URL:s will be dead by the time the paper leaves the presses. Furthermore, those same papers, the ones that won't immediately fall into oblivion, will be read for another 5-10-20 years, even in CS/CE. The average "web" year quite frankly does not cut it. So much for your "as long as it lives".

Furthermore, the research community of course won't stop using the web, or the internet, they were the ones that created it, remember? Private citizens weren't even allowed in. In the case of the web, the communication of scientific research results were the sole motivation in the first place. The first web page ever communicated scientific results.

No, the research community, knows full well, what the technology is capable of, and today, quite frankly, said technology is not capable of providing a source for scientific references that is, accurate, reliable, long lived, etc. etc. The internet/web may be fine for other (early) forms of scientific communication, but this isn't one of them.

This may well change, and in fact said researchers are working on the problem right now, see for instance, Ross Anderson's "Eternity service", but today, we are simply not there.

That's all I have to say to you, Mr. Nilaad, on the subject. No doubt, you'll feel the urge keep on spewing.

Stefan,
--
Stefan Axelsson Chalmers University of Technology
sax@rmovt.rply.ce.chalmers.se Dept. of Computer Engineering
(Remove "rmovt.rply" to send mail.)

Subject: Re: Memo to the Amateur Cipher Designer
Date: Mon, 16 Nov 1998 22:32:36 GMT
From: "Joseph K. Nilaad" <jknilaad@xoommail.com>
Message-ID: <3650A804.19D6@xoommail.com>
References: <72k7r5mju1@nyheter.chalmers.se>
Newsgroups: sci.crypt
Lines: 51

Stefan Axelsson wrote:
>
> Look, the average time from research to publication in a refereed
> journal today is two years. Many/most of those URL:s will be dead by
Yes, I know that. By the time it is published, the matterial may not be
applicable 2-3 years later.

> the time the paper leaves the presses. Furthermore, those same papers,
> the ones that won't immediately fall into oblivion, will be read for
> another 5-10-20 years, even in CS/CE. The average "web" year quite
This is true for some other fields. In computer? I doubt it.

> frankly does not cut it. So much for your "as long as it lives".
My point is that so what, if it's last less than one second. if you
read the posted URL and get the ideas from it, then you should refer it
unless you want to tell the world that it's your idea. Will you?

Correct me if I'm wrong, other than common knowledge, we should give
credit to those whom we got some ideas from. URL or not URL, hardcopy
or not.

>
> Furthermore, the research community of course won't stop using the
> web, or the internet, they were the ones that created it, remember?
> Private citizens weren't even allowed in. In the case of the web, the
> communication of scientific research results were the sole motivation
> in the first place. The first web page ever communicated scientific
> results.
Of course not. This is one of the most efficient way to communicate
with colleagues. If I get some ideas from you through this means and
say the ideas are mine, you won't like that.
>
[snip]
> reliable, long lived, etc. etc. The internet/web may be fine for other
> (early) forms of scientific communication, but this isn't one of them.
Why it was OK then, but not now. Give me some scientific reason(s).

> This may well change, and in fact said researchers are working on the
> problem right now, see for instance, Ross Anderson's "Eternity
> service", but today, we are simply not there.
The obvious problems that I can see with URL are authenticateion and
short live. See my reply to Bruce Scheier in sci.crypt for detail.

>
> That's all I have to say to you, Mr. Nilaad, on the subject. No doubt,

> you'll feel the urge keep on spewing.
Look, just because publishing via URL is relatively short life comparing
with hard copies, it doesn't mean we should not give publishers their
credits. Unless, if you think that swindling someone's idea is OK.
That would really gag me.

Subject: Re: Memo to the Amateur Cipher Designer
Date: Tue, 17 Nov 1998 01:07:54 -0600
From: jgfunj@EnqvbSerrGrknf.pbz (W T Shaw)
Message-ID: <jgfunj-1711980107540001@dialup163.itexas.net>
References: <3650A804.19D6@xoommail.com>
Newsgroups: sci.crypt
Lines: 27

In article <3650A804.19D6@xoommail.com>, jknilaad@xoommail.com wrote:
>
> Correct me if I'm wrong, other than common knowledge, we should give
> credit to those whom we got some ideas from. URL or not URL, hardcopy
> or not.
>
If you can give credit, fine. It may be that you do not remember reading
something, but use the idea anyway. In scanning lots of material, the end
use of tidbits is not always obvious, and taking a note on where
everything comes from is next to impossible. It is another case where
demanding full footnotes may sound good, but it is next to impossible.
Being able to scan the full archives should help.

....
> Look, just because publishing via URL is relatively short life comparing
> with hard copies, it doesn't mean we should not give publishers their
> credits. Unless, if you think that swindling someone's idea is OK.
> That would really gag me.

It may not be intentional, and sometimes people do have convergent
thoughts. Try not to get too choked up about it. The only honest thing
to do is to quote references when you know them, and not when you don't.
--

Your future is ahead of you.--Thomas E. Dewey

Decrypt with ROT13 to get correct email address.

Subject: Re: Memo to the Amateur Cipher Designer
Date: 17 Nov 1998 11:49:49 GMT
From: sax@rmovt.rply.ce.chalmers.se (Stefan Axelsson)
Message-ID: <72rnst$t6t$1@nyheter.chalmers.se>
References: <3650A804.19D6@xoommail.com>
Newsgroups: sci.crypt
Lines: 40

In article <3650A804.19D6@xoommail.com>,
Joseph K. Nilaad <jknilaad@xoommail.com> wrote:

>Look, just because publishing via URL is relatively short life comparing
>with hard copies, it doesn't mean we should not give publishers their
>credits.

As I said in my original posting, the one which wound you up to no
end, there are references, and there are references. If your only
motivation for including a reference is to acknowledge someone else's,
idea, then the name of said person would (in general) do nicely. If
you include a URL, it should be with the knowledge that it is/will
become useless to the reader in a very short period of time. Read
Patrick Juola's post in this thread, he's put it very succinctly.

Now, when it comes to your time frame, believe me when I say that *I*
read plenty of papers that are 5, 10 (even) 20 years old (and so do my
peers btw), and computer security, which is my field, is moving as
fast as the rest of them. 6 months just won't cut it.

And this is just one of the problems, others being lack of peer
review, authenticity etc. etc.

The main use of electronic communication in the research community
today is to learn of research results, (what's everybody else up to?),
down load papers already published elsewhere (while waiting for them,
or deciding whether to bother, to arrive from the library), and
communicating with peers around the world to do research, write about
it, and review what other's have written. (And of course the telephone
and fax are not excluded from the means of electronic communication.)

Read my lips: For the peer reviewed publication of research results,
today, a URL, just will not cut it, and this will be true for a long
time to come (when counting web years at least)

Stefan,
--
Stefan Axelsson Chalmers University of Technology
sax@rmovt.rply.ce.chalmers.se Dept. of Computer Engineering
(Remove "rmovt.rply" to send mail.)

Subject: Re: Memo to the Amateur Cipher Designer
Date: Thu, 12 Nov 1998 22:00:49 GMT
From: ritter@io.com (Terry Ritter)
Message-ID: <364b5a73.14615148@news.io.com>
References: <72bj3tbbk1@nyheter.chalmers.se>
Newsgroups: sci.crypt
Lines: 47

On 11 Nov 1998 08:50:05 GMT, in <72bj3tbbk1@nyheter.chalmers.se>, in sci.crypt sax@rmovt.rply.ce.chalmers.se (Stefan Axelsson) wrote:

>[...]
>The problem of what to do
>with URL:s is a somewhat debated topic in academic circles today, and
>no real consensus has been reached. However, the majority of
>researchers recognise that there are difficult, and fundamental
>problems with referring to URL:s, or other forms of transient
>communication.

Problems with URL's include the fact that their contents change over time, and that the URL's themselves may change or die. Still, as opposed to having *no* introductions to the literature, URL's can make a contribution.

>The requirements for a "good" scientific paper, whatever that may be,
>today, I would say, would exclude even references to obscure
>publications, where they not crucial to the treatment of the subject.

This addresses the *convenience* of Science to the reader. But it ignores the *responsibility* of the author and the *requirement* of scientific publication to acknowledge the previous work, the source of the inspiration (rarely is any work completely original). If that previous work came in a private letter, so be it.

>A newsgroup posting, I would say, is (almost) right up there with
>"personal conversation with N.N.", dejanews or not. It detracts from
>the scientific value of the paper, it doesn't add to it!

A News posting has a "fixed expression" as of a given date, and a message-ID that identifies it particularly. It is also available in libraries, and all this makes it a "publication" for legal purposes.

Nor is DejaNews the only News archive (I know of Reference.com; maybe somebody will know of others which do not use the DejaNews engine). And News articles *can* be given a URL which has meaning, simply because the News article has a particular fixed expression.

Terry Ritter ritter@io.com http://www.io.com/~ritter/
Crypto Glossary http://www.io.com/~ritter/GLOSSARY.HTM

Subject: Re: Memo to the Amateur Cipher Designer
Date: 14 Nov 1998 15:18:53 GMT
From: sax@rmovt.rply.ce.chalmers.se (Stefan Axelsson)
Message-ID: <72k70tksj1@nyheter.chalmers.se>
References: <364b5a73.14615148@news.io.com>
Newsgroups: sci.crypt
Lines: 36

In article <364b5a73.14615148@news.io.com>, Terry Ritter <ritter@io.com> wrote:

>This addresses the *convenience* of Science to the reader. But it
>ignores the *responsibility* of the author and the *requirement* of
>scientific publication to acknowledge the previous work, the source of
>the inspiration (rarely is any work completely original). If that
>previous work came in a private letter, so be it.

No, that's not what I meant. Note my comment that not all references are created equal. In either case, if you *build* your argument on something you reference, then this reference should be reliably available to your peers.

>A News posting has a "fixed expression" as of a given date, and a
>message-ID that identifies it particularly. It is also available in
>libraries, and all this makes it a "publication" for legal purposes.

Not in any library I know of... Now, I won't go into the american legal definition of "publication". Save to say that in (most) academic circles, a web page, or a news article does not a publication make. And this, I would add, for good, already stated, reasons.

The web/internet is an outstanding medium for many forms of scientific communication, and indeed that was the reason *) the underlying technology was created in the first place, but for references in peer reviewed publications, it leaves too much to be desired.

*) Sole reason in the case of the web, less so in the case of the internet.

My last post in this subthread. Feel free to have the last word.

Stefan,
--
Stefan Axelsson Chalmers University of Technology
sax@rmovt.rply.ce.chalmers.se Dept. of Computer Engineering
(Remove "rmovt.rply" to send mail.)

Subject: Re: Memo to the Amateur Cipher Designer
Date: Wed, 11 Nov 1998 16:52:29 GMT
From: schneier@counterpane.com (Bruce Schneier)
Message-ID: <3649c053.4536194@news.visi.com>
References: <36494616.1500C031@stud.uni-muenchen.de>
Newsgroups: sci.crypt
Lines: 40

On Wed, 11 Nov 1998 09:08:54 +0100, Mok-Kong Shen
<mok-kong.shen@stud.uni-muenchen.de> wrote:
>Someone pointed out to me that I could have saved more bandwidth if
>I had paid attention to the following fact:
>
> In the academically 'peer reviewed' paper by J. Kelsey, B.
> Schneier, D. Wagner, C. Hall, Cryptanalytic Attacks on Pseudo-
> random Number Generators (in S. Vaudenay (ed), Fast Software
> Encryption, Springer, 1998), there is the following entry in
> the references:
>
> [Koc95] P. Kocher, post to sci.crypt Internet newsgroup (message-
> ID pckDIr4Ar.L4zQnetcom.com), 4 Dec 1995
>
>This clearly refutes the claim that references to newsgroup articles
>are unworthy of good scientific papers and shows up the existence
>of some more logical inconsistencies than I had been able to uncover.
>
>It is noteworthy that in the same paper there are also references to
>Web page URL and to ftp URL. So these are also valid references
>for good scientific papers. Should patent publications be less
>worthy of being cited scientifically?? (In scientific publications
>one sometimes even see such references as 'Privte communiction from
>XXX'. I personally consider these to be rather useless since the
>reader has practically no possibility to access these communications.)

Fascinating. And I consider myself avant guard by citing email
messages, URLs, and patents in my papers. I take some flak for it,
but I do it anyway. Most others don't bother, although it is much
more common in computer security circles to add a URL if a paper
appears on a website in addition to a proceedings or journal.

Whatever. I'm not sure what the discussion is about anymore, and I
don't really want to bother figuring it out.

Bruce
****
Bruce Schneier, President, Counterpane Systems Phone: 612-823-1098
101 E Minnehaha Parkway, Minneapolis, MN 55419 Fax: 612-823-1590
Free crypto newsletter. See: http://www.counterpane.com

Subject: Re: Memo to the Amateur Cipher Designer
Date: Wed, 11 Nov 1998 19:36:11 +0100
From: Mok-Kong Shen <mok-kong.shen@stud.uni-muenchen.de>
Message-ID: <3649D91B.36B84298@stud.uni-muenchen.de>
References: <3649c053.4536194@news.visi.com>
Newsgroups: sci.crypt
Lines: 46

Bruce Schneier wrote:
>
> On Wed, 11 Nov 1998 09:08:54 +0100, Mok-Kong Shen
> <mok-kong.shen@stud.uni-muenchen.de> wrote:
> >Someone pointed out to me that I could have saved more bandwidth if
> >I had paid attention to the following fact:
> >
> > In the academically 'peer reviewed' paper by J. Kelsey, B.
> > Schneier, D. Wagner, C. Hall, Cryptanalytic Attacks on Pseudo-
> > random Number Generators (in S. Vaudenay (ed), Fast Software
> > Encryption, Springer, 1998), there is the following entry in
> > the references:
> >
> > [Koc95] P. Kocher, post to sci.crypt Internet newsgroup (message-
> > ID pckDIr4Ar.L4zQnetcom.com), 4 Dec 1995
> >
> >This clearly refutes the claim that references to newsgroup articles
> >are unworthy of good scientific papers and shows up the existence
> >of some more logical inconsistencies than I had been able to uncover.
> >
> >It is noteworthy that in the same paper there are also references to
> >Web page URL and to ftp URL. So these are also valid references
> >for good scientific papers. Should patent publications be less
> >worthy of being cited scientifically?? (In scientific publications
> >one sometimes even see such references as 'Privte communiction from
> >XXX'. I personally consider these to be rather useless since the
> >reader has practically no possibility to access these communications.)

>
> Fascinating. And I consider myself avant guard by citing email
> messages, URLs, and patents in my papers. I take some flak for it,
> but I do it anyway. Most others don't bother, although it is much
> more common in computer security circles to add a URL if a paper
> appears on a website in addition to a proceedings or journal.
>
> Whatever. I'm not sure what the discussion is about anymore, and I
> don't really want to bother figuring it out.

To be an avantguard is one thing yet not to mention a relevant
fact that a scientist himself is MOST familiar with (because it
concerns HIS own writing) in a scientific discussion and that obviously
consciously (unless one suffers from the blackout-syndrom of
some politicians) and thus misleading others to waste bandwidth and
above that also to accuse these others having wasted bandwidth is
certainly entirely another matter!

M. K. Shen

Subject: Re: Memo to the Amateur Cipher Designer
Date: Wed, 11 Nov 1998 18:50:02 GMT
From: schneier@counterpane.com (Bruce Schneier)
Message-ID: <3649dc0c.11634433@news.visi.com>
References: <3649D91B.36B84298@stud.uni-muenchen.de>
Newsgroups: sci.crypt
Lines: 19

On Wed, 11 Nov 1998 19:36:11 +0100, Mok-Kong Shen
<mok-kong.shen@stud.uni-muenchen.de> wrote:
>To be an avantguard is one thing yet not to mention a relevant
>fact that a scientist himself is MOST familiar with (because it
>concerns HIS own writing) in a scientific discussion and that obviously
>consciously (unless one suffers from the blackout-syndrom of
>some politicians) and thus misleading others to waste bandwidth and
>above that also to accuse these others having wasted bandwidth is
>certainly entirely another matter!

Yeah. Sure. You're right. Whatever. I apologise for misleading you
(delierately, it seems) can causing you to waste bandwidth and then to
accuse you of wasting bandwidth.

Bruce
****
Bruce Schneier, President, Counterpane Systems Phone: 612-823-1098
101 E Minnehaha Parkway, Minneapolis, MN 55419 Fax: 612-823-1590
Free crypto newsletter. See: http://www.counterpane.com

Subject: Re: Memo to the Amateur Cipher Designer
Date: Wed, 11 Nov 1998 16:52:53 -0600
From: jgfunj@EnqvbSerrGrknf.pbz (W T Shaw)
Message-ID: <jgfunj-1111981652530001@dialup164.itexas.net>
References: <3649c053.4536194@news.visi.com>
Newsgroups: sci.crypt
Lines: 15

In article <3649c053.4536194@news.visi.com>, schneier@counterpane.com
(Bruce Schneier) wrote:
>
> Fascinating. And I consider myself avant guard by citing email
> messages, URLs, and patents in my papers. I take some flak for it,
> but I do it anyway. Most others don't bother, although it is much
> more common in computer security circles to add a URL if a paper
> appears on a website in addition to a proceedings or journal.
>
You have that in common with Ritter, as I recall.
--

The public is harder to steamroller than some might think.

Decrypt with ROT13 to get correct email address.

Subject: Re: Memo to the Amateur Cipher Designer
Date: Wed, 11 Nov 1998 22:31:47 GMT
From: schneier@counterpane.com (Bruce Schneier)
Message-ID: <364a1014.24956416@news.visi.com>
References: <jgfunj-1111981652530001@dialup164.itexas.net>
Newsgroups: sci.crypt
Lines: 26

On Wed, 11 Nov 1998 16:52:53 -0600, jgfunj@EnqvbSerrGrknf.pbz (W T
Shaw) wrote:

>In article <3649c053.4536194@news.visi.com>, schneier@counterpane.com
>(Bruce Schneier) wrote:
>>
>> Fascinating. And I consider myself avant guard by citing email
>> messages, URLs, and patents in my papers. I take some flak for it,
>> but I do it anyway. Most others don't bother, although it is much
>> more common in computer security circles to add a URL if a paper

>> appears on a website in addition to a proceedings or journal.
>>
>You have that in common with Ritter, as I recall.

There are others, too. Currently the academic community is still trying to figure out how to handle URL references. The problem is that they are not stagnant, as references usually are. That is, if I reference a URL, and someone reads my paper two years later and looks at the same URL, they may not see what I saw. This problem does not exist with journals and conference proceedings.

Bruce
****
Bruce Schneier, President, Counterpane Systems Phone: 612-823-1098
101 E Minnehaha Parkway, Minneapolis, MN 55419 Fax: 612-823-1590
Free crypto newsletter. See: http://www.counterpane.com

Subject: Re: Memo to the Amateur Cipher Designer
Date: Thu, 12 Nov 1998 11:43:48 -0600
From: jgfunj@EnqvbSerrGrknf.pbz (W T Shaw)
Message-ID: <jgfunj-1211981143480001@207.22.198.202>
References: <364a1014.24956416@news.visi.com>
Newsgroups: sci.crypt
Lines: 20

In article <364a1014.24956416@news.visi.com>, schneier@counterpane.com (Bruce Schneier) wrote:
>
>..Currently the academic community is still
> trying to figure out how to handle URL references. The problem is
> that they are not stagnant, as references usually are. That is, if I
> reference a URL, and someone reads my paper two years later and looks
> at the same URL, they may not see what I saw. This problem does not
> exist with journals and conference proceedings.
>
Fair use should mean that you could post the reference if it disappeared.
Important things change from what is printed in journals and books too,
job titles, mailing addresses and phone numbers. Actual technical
mistakes are rather hard to reverse as well in fixed media; note the
increased leaning on the web for current updates.
--

Your future is ahead of you.--Thomas E. Dewey

Decrypt with ROT13 to get correct email address.

Subject: Re: Memo to the Amateur Cipher Designer
Date: Thu, 12 Nov 1998 19:16:36 GMT
From: schneier@counterpane.com (Bruce Schneier)
Message-ID: <364c3360.14791002@news.visi.com>
References: <jgfunj-1211981143480001@207.22.198.202>
Newsgroups: sci.crypt
Lines: 47

On Thu, 12 Nov 1998 11:43:48 -0600, jgfunj@EnqvbSerrGrknf.pbz (W T Shaw) wrote:

>In article <364a1014.24956416@news.visi.com>, schneier@counterpane.com
>(Bruce Schneier) wrote:
>>
>>..Currently the academic community is still
>> trying to figure out how to handle URL references. The problem is
>> that they are not stagnant, as references usually are. That is, if I
>> reference a URL, and someone reads my paper two years later and looks
>> at the same URL, they may not see what I saw. This problem does not
>> exist with journals and conference proceedings.
>>
>Fair use should mean that you could post the reference if it disappeared.
>Important things change from what is printed in journals and books too,
>job titles, mailing addresses and phone numbers. Actual technical
>mistakes are rather hard to reverse as well in fixed media; note the
>increased leaning on the web for current updates.

Of course job titles, mailing addresses, and phone numbers change. That's not the issue.

The issue is that the actual reference may change. That is, in a printed journal the page I look at when I write the paper is the same page you look at when you check my reference. It is the same text, the same mailing address, the same everything. This is how conventional references work.

URL references do not work this way. The page I look at when I write my paper may or may not be the same page you look at when you check my reference. And neither of us has any way of knowing what will change in the future or what has changed in the past.

This is not a complaint, criticism, or anything like that. I am not saying that URLs are not valid references. I am not saying that URLs are necessarily less valuable than articles. I am not saying that academics ignore URLs.

All I am saying is that there is an essential characteristic that is different, and the academic community is still trying to figure out how to handle that difference.

Bruce
****
Bruce Schneier, President, Counterpane Systems Phone: 612-823-1098
101 E Minnehaha Parkway, Minneapolis, MN 55419 Fax: 612-823-1590
Free crypto newsletter. See: http://www.counterpane.com

Subject: Re: Memo to the Amateur Cipher Designer
Date: Thu, 12 Nov 1998 15:05:19 -0500
From: Tim Bass <"nobody"@silkroad.com (No Spam Please)>
Message-ID: <xiH22.2457$fl.17272491@audrey2.cais.com>
References: <364c3360.14791002@news.visi.com>
Newsgroups: sci.crypt
Lines: 67

> The issue is that the actual reference may change. That is, in a
> printed journal the page I look at when I write the paper is the same
> page you look at when you check my reference.

Absolutely!!!

The primary purpose of technical writing, publishing, etc. is to communicate knowledge which is consistant, accurate, and is grounded in historical events and facts.

With the current state of the network, it is quite unprofessional to reference URLS. URLS are not static. They can change; they do change. The information content may change. In fact, I can easily modify my apache server to provide a completely different paper (URL) to the network user based on ip address, domain, or userid/password.

The same info that Sally, Ted, and Alice read which was written by Joe must be the same.

In addition, historians in some future year looking for Mr. Bass' papers (if he was alive in the year 2050) may not be able to find http://www.silkroad.com/papers/. They may, however, have luck with a CDROM database of a statically reviewed paper, or paper journals.

Could we please discuss the technical aspects of crypto and cryptanalysis and work constructively on topics which are important? Instead of shouting opinions and picking apart other spelling or grammer, how about we work in building a new algorithm together and do the cryptanalysis together and have some fun!

I don't know about you, but I think cryptanalysis is a great topic, very interesting, and fun. It is not important to me 'who gets credit for what' and 'who knows more about x than y'. Everyone has something to offer and if given a chance and encouragement, many will become experts and scholars.

Unfortunately, the interesting technical threads 'just die '; while the ' mud slinging ' threads with harsh speech and strong ungrounded opinions live on. It's it interesting how little we humans have evolved! We have WWW servers, 300 MHZ processors, LCD touch screens, and with all this, we just throw rocks and mud at each other.

Let's have some technical fun!

-Tim
--
Tim Bass
Principal Consultant, Systems Engineering
The Silk Road Group, Ltd.
Tel: (703) 222-4243
Fax: (703) 222-7320
EMail: bass (at symbol) silkroad (.) com
http://www.silkroad.com/
http://www.silkroad.com/consulting/technical.html

Subject: Re: Memo to the Amateur Cipher Designer
Date: Thu, 12 Nov 1998 22:01:02 GMT
From: ritter@io.com (Terry Ritter)
Message-ID: <364b5a95.14648894@news.io.com>
References: <364a1014.24956416@news.visi.com>

Newsgroups: sci.crypt
Lines: 41

On Wed, 11 Nov 1998 22:31:47 GMT, in
<364a1014.24956416@news.visi.com>, in sci.crypt
schneier@counterpane.com (Bruce Schneier) wrote:

>[...]
>Currently the academic community is still
>trying to figure out how to handle URL references. The problem is
>that they are not stagnant, as references usually are. That is, if I
>reference a URL, and someone reads my paper two years later and looks
>at the same URL, they may not see what I saw. This problem does not
>exist with journals and conference proceedings.

This concept is often called "fixed expression," and is fundamental to
the concept of "publication": When content is changed, which version
is "the" publication?

But this happens with books as well, and I often find myself dealing
with an different version of a well-known work than someone else is
quoting. I handle this, when necessary, by going to the library and
getting "the" reference. The problem with the Web is that the older
publications actually *disappear*, and that *is* a problem.

One thing we *could* do is to use "document information" to give us a
particular file date, but that is not going to be much use when others
have no access to the older document. It also can change without the
content changing, for example, when the ISP moves to a new machine.
So I guess that most Web URL's need to be seen as "additional
information," not central to an argument. But Web URL's *can* be an
appropriate way to further identify the source of an idea which has
been expanded differently.

Note that the "fixed expression" issue does not occur with News
articles, which do have a particular expression fixed by a particular
message-ID.

Terry Ritter ritter@io.com http://www.io.com/~ritter/
Crypto Glossary http://www.io.com/~ritter/GLOSSARY.HTM

Subject: Re: Memo to the Amateur Cipher Designer
Date: Fri, 13 Nov 1998 15:12:13 +0100
From: Mok-Kong Shen <mok-kong.shen@stud.uni-muenchen.de>
Message-ID: <364C3E3D.88571908@stud.uni-muenchen.de>
References: <364a1014.24956416@news.visi.com>
Newsgroups: sci.crypt
Lines: 25

Bruce Schneier wrote:

> There are others, too. Currently the academic community is still
> trying to figure out how to handle URL references. The problem is
> that they are not stagnant, as references usually are. That is, if I
> reference a URL, and someone reads my paper two years later and looks
> at the same URL, they may not see what I saw. This problem does not
> exist with journals and conference proceedings.

Informations on the internet, in particular Web, is getting archived.
I personally have not yet made use of this resource but here is
the URL:

http://www.archive.org/

and here is it what it claims anyway:

> The Archive will provide historians, researchers, scholars,
> and others access to this vast collection of data (reaching
> ten terabytes), and ensure the longevity of this information.

Assuming that this archive (or similar archives in future) does a
good job, the said problem should disappear.

M. K. Shen

Subject: Re: Memo to the Amateur Cipher Designer
Date: 13 Nov 1998 09:26:04 -0500
From: juola@mathcs.duq.edu (Patrick Juola)
Message-ID: <72hfhs$fv9$1@quine.mathcs.duq.edu>
References: <364C3E3D.88571908@stud.uni-muenchen.de>
Newsgroups: sci.crypt
Lines: 40

In article <364C3E3D.88571908@stud.uni-muenchen.de>,
Mok-Kong Shen <mok-kong.shen@stud.uni-muenchen.de> wrote:
>Bruce Schneier wrote:
>
>> There are others, too. Currently the academic community is still

>> trying to figure out how to handle URL references. The problem is
>> that they are not stagnant, as references usually are. That is, if I
>> reference a URL, and someone reads my paper two years later and looks
>> at the same URL, they may not see what I saw. This problem does not
>> exist with journals and conference proceedings.
>
>Informations on the internet, in particular Web, is getting archived.
>I personally have not yet made use of this resource but here is
>the URL:
>
>   http://www.archive.org/
>
>and here is it what it claims anyway:
>
>   The Archive will provide historians, researchers, scholars,
>   and others access to this vast collection of data (reaching
>   ten terabytes), and ensure the longevity of this information.
>
>Assuming that this archive (or similar archives in future) does a
>good job, the said problem should disappear.

Absolutely. But that's a *HUGE* assumption -- the more so as the
volume of information on the web at any one time is, to a first
approximation, several percent of the total hard drive capacity
manufactured and in service.

Think of it this way -- assuming the average document half-life
is about six months (which I pulled out of thin air, but seems
about right), then you'll need to buy the entire Web in terms of
disk capacity EVERY YEAR.

I don't think it's prudent to assume that someone will make that
investment of time and money.

        -kitten

Subject: Re: Memo to the Amateur Cipher Designer
Date: 18 Nov 1998 14:09:39 GMT
From: clvisser@gene.wins.uva.nl (Coen L.S. Visser)
Message-ID: <72ukf3$938$1@nbox.wins.uva.nl>
References: <72hfhs$fv9$1@quine.mathcs.duq.edu>
Newsgroups: sci.crypt
Lines: 24

juola@mathcs.duq.edu (Patrick Juola) writes:
>Mok-Kong Shen <mok-kong.shen@stud.uni-muenchen.de> wrote:
>>Bruce Schneier wrote:

Bruce> Currently the academic community is still
Bruce> trying to figure out how to handle URL references. The problem is
Bruce> that they are not stagnant, as references usually are.

Mok-Kong> Informations on the internet, in particular Web, is getting archived.
Mok-Kong> Assuming that this archive (or similar archives in future) does a
Mok-Kong> good job, the said problem should disappear.

Patrick> Absolutely. But that's a *HUGE* assumption

I think the author making the reference should be responsible for archiving
the particular web page in case the original reference becomes invalid.
That creates of course copyright issues.
Furthermore there is the dilemma which link should be printed as the reference.
The original web page or the author's copy (with a reference from the
author's page to the original of course).

Regards,

        Coen

Subject: Re: Memo to the Amateur Cipher Designer
Date: 18 Nov 1998 09:52:12 -0500
From: juola@mathcs.duq.edu (Patrick Juola)
Message-ID: <72umus$pc6$1@quine.mathcs.duq.edu>
References: <72ukf3$938$1@nbox.wins.uva.nl>
Newsgroups: sci.crypt
Lines: 36

In article <72ukf3$938$1@nbox.wins.uva.nl>,
Coen L.S. Visser <clvisser@gene.wins.uva.nl> wrote:
>juola@mathcs.duq.edu (Patrick Juola) writes:
>>Mok-Kong Shen <mok-kong.shen@stud.uni-muenchen.de> wrote:
>>>Bruce Schneier wrote:
>
>Bruce> Currently the academic community is still
>Bruce> trying to figure out how to handle URL references. The problem is
>Bruce> that they are not stagnant, as references usually are.
>
>Mok-Kong> Informations on the internet, in particular Web, is getting archived.
>Mok-Kong> Assuming that this archive (or similar archives in future) does a
>Mok-Kong> good job, the said problem should disappear.
>
>Patrick> Absolutely. But that's a *HUGE* assumption

>
>I think the author making the reference should be responsible for archiving
>the particular web page in case the original reference becomes invalid.

I think that's about the fourth most unreasonable assertion I've heard
in my life. In the second place, it's no more likely that I will be
findable in three years than the original author of the URL under
discussion.

And in the first place, it's an unreasonable burden to place on the
author. I neither have disk space, money, time, or (most importantly)
interest in archiving every URL I've ever found useful on the off-chance
that someone might call upon me to produce my references. The whole
point of the reference scheme is to point the reader to someplace
where s/he can be reasonably expected to find background material
relevant to my document. To expect the author to do the work of
a librarian as well as an author is completely out of line. Why
don't you expect me to hand-deliver copies of my manuscript as well,
just in case the postal trucks break down?

-kitten

Subject: Re: Memo to the Amateur Cipher Designer
Date: 23 Nov 1998 01:33:53 GMT
From: clvisser@gene.wins.uva.nl (Coen L.S. Visser)
Message-ID: <73ae21oo1@nbox.wins.uva.nl>
References: <72umus$pc6$1@quine.mathcs.duq.edu>
Newsgroups: sci.crypt
Lines: 59

juola@mathcs.duq.edu (Patrick Juola) writes:
>Coen L.S. Visser <clvisser@gene.wins.uva.nl> wrote:

>>I think the author making the reference should be responsible for archiving
>>the particular web page in case the original reference becomes invalid.

>I think that's about the fourth most unreasonable assertion I've heard
>in my life.

Ok, I want to know: what are the other three? But seriously ;-)

>In the second place, it's no more likely that I will be
>findable in three years than the original author of the URL under
>discussion.

There is always the risk that you change ISP or something like that.
But if you write books for a living and you have a web page, I believe
the chances are quite high that your (new) web page can be found.

>And in the first place, it's an unreasonable burden to place on the
>author. I neither have disk space, money, time, or (most importantly)
>interest in archiving every URL I've ever found useful on the off-chance
>that someone might call upon me to produce my references.

It is possible to make a distinction between an important reference
(one whose web pages you mirror) and an unimportant web reference which
you just mention.

>The whole point of the reference scheme is to point the reader to someplace
>where s/he can be reasonably expected to find background material
>relevant to my document.

Unfortunately, information on the net is much more volatile than that in
printed media. There are some big discussions about the severe amnesia
the internet suffers from. There is no need to make it worse. If you
write something that you think is still important in five years and it
contains a vital web page reference you would do well to preserve it.
And if your paper is outdated in a year or so, well why bother mirroring
all the web pages you refer to.

>To expect the author to do the work of a librarian as well as an author is
>completely out of line.

My statement that an author is responsible for his/her references was
a bit unjust I think now. Of course not all the work of mirroring should
come on the shoulders of authors. There are many ways how to manage
a mirroring scheme. A research institute could create a reference repository.
If it is your Big Book (TM), you could harass your publisher to mirror the
documents that you find important. Many publishers already make errata to
"their" books available in electronic form.

Regards,

Coen Visser

NB

I've posted my opinion in this thread twice because my posting disappeared
from our local news server. Apologies for the waste of bandwidth.

Subject: Re: Memo to the Amateur Cipher Designer
Date: Fri, 13 Nov 1998 14:23:36 GMT

From: "Joseph K. Nilaad" <jknilaad@xoommail.com>
Message-ID: <364C40E8.1D24@xoommail.com>
References: <364a1014.24956416@news.visi.com>
Newsgroups: sci.crypt
Lines: 28

Bruce Schneier wrote:
>
> There are others, too. Currently the academic community is still
> trying to figure out how to handle URL references. The problem is
> that they are not stagnant, as references usually are. That is, if I
> reference a URL, and someone reads my paper two years later and looks
> at the same URL, they may not see what I saw. This problem does not
> exist with journals and conference proceedings.
This is just a thought of handling referred URL documents:
If a document has references from any URL, those URL referrences must be
electronically signed. By doing this, we can have authentic documents.
In addition the author of the document must copy the whole URL documents
and keep them as referrences. In case of any conflict, the
electronically copied documents can be used as proofs. For signing the
document, URL address must always be included with the original document
owner's signature.

However, I still see some potential problem.
1. Which method is used to provide authentication?
2. Who will keep the database of disappeared URL?

Are there any other ideas?

Joseph K. Nilaad
Nature is simple and beautiful...
Life is too short to appreciate it all...

Subject: Re: Memo to the Amateur Cipher Designer
Date: Tue, 17 Nov 1998 09:05:00 +0100
From: Mok-Kong Shen <mok-kong.shen@stud.uni-muenchen.de>
Message-ID: <36512E2C.CB9FB39B@stud.uni-muenchen.de>
References: <364C40E8.1D24@xoommail.com>
Newsgroups: sci.crypt
Lines: 32

Joseph K. Nilaad wrote:

> This is just a thought of handling referred URL documents:
> If a document has references from any URL, those URL referrences must be
> electronically signed. By doing this, we can have authentic documents.
> In addition the author of the document must copy the whole URL documents
> and keep them as referrences. In case of any conflict, the
> electronically copied documents can be used as proofs. For signing the
> document, URL address must always be included with the original document
> owner's signature.
>
> However, I still see some potential problem.
> 1. Which method is used to provide authentication?
> 2. Who will keep the database of disappeared URL?

I see rather the same problems with the originals of published
Web pages. Due to the inherent insecurity of the internet (see faked
mail addresses, for example) there is the question of authentication
of the author. Of course, paper publications actually also have
the same problem. But I suppose the problem really becomes serious
with Web publications. As to disappeared URL the archives I
reported in another post should remove the problem, at least in
principle. However, the volume of informations on the internet
is expanding (super-)exponentially. There is the question of
economy of storing them, if one simply archives everything. One
way I could imagine is for the archives to charge some fees for
the archiving rather than offering free service. Another problms is
that there are more often updates to Web pages than to printed stuffs.
One could use the technique of storing the differences like in version
management of software. But clearly there is some tough problem here.

M. K. Shen

Subject: Re: Memo to the Amateur Cipher Designer
Date: 17 Nov 1998 12:00:56 GMT
From: sax@rmovt.rply.ce.chalmers.se (Stefan Axelsson)
Message-ID: <72roho$18$1@nyheter.chalmers.se>
References: <36512E2C.CB9FB39B@stud.uni-muenchen.de>
Newsgroups: sci.crypt
Lines: 43

In article <36512E2C.CB9FB39B@stud.uni-muenchen.de>,
Mok-Kong Shen <mok-kong.shen@stud.uni-muenchen.de> wrote:

>I see rather the same problems with the originals of published
>Web pages. Due to the inherent insecurity of the internet (see faked
>mail addresses, for example) there is the question of authentication
>of the author. Of course, paper publications actually also have
>the same problem.

Not really. You see, this is actually a model of trust in action. Some of the societies that peer review scientific work have the highest reputation, others do not. While there's the occasional slip-up this in general works very well. (Much of it, of course, hinges on the ability of the readers of said peer reviewed work to be able to check the references...) The IEEE stands behind the printed copy, and they are able to do so with some authority, and confidence.

Now, if the IEEE for example where to say, OK, to h*ll with the dead trees, let there be business as usual, but on the web instead, then of course, (almost) all that which is the IEEE would transfer to the electronic medium, and little would have to change.

The situation with everyone "publishing" their material is so far removed from this that I don't know where to start. Suffice it to say, that as is common, it is not (mainly) a question of technology, but one of the human element. What do we want, and how do we accomplish that?

>but clearly there is some tough problem here.
>M. K. Shen

Amen.

P.S. I've continued the discussion on a number of occasions, when I've promised not to, and this is getting so far removed from sci.crypt, that I suggest that the interested parties (myself included) take it to email.

Stefan,
--
Stefan Axelsson Chalmers University of Technology
sax@rmovt.rply.ce.chalmers.se Dept. of Computer Engineering
(Remove "rmovt.rply" to send mail.)

Subject: Re: Memo to the Amateur Cipher Designer
Date: Tue, 17 Nov 1998 13:38:07 +0100
From: Mok-Kong Shen <mok-kong.shen@stud.uni-muenchen.de>
Message-ID: <36516E2F.FB265F4A@stud.uni-muenchen.de>
References: <72roho$18$1@nyheter.chalmers.se>
Newsgroups: sci.crypt
Lines: 26

Stefan Axelsson wrote:
>
> In article <36512E2C.CB9FB39B@stud.uni-muenchen.de>,
> Mok-Kong Shen <mok-kong.shen@stud.uni-muenchen.de> wrote:
>
> >I see rather the same problems with the originals of published
> >Web pages. Due to the inherent insecurity of the internet (see faked
> >mail addresses, for example) there is the question of authentication
> >of the author. Of course, paper publications actually also have
> >the same problem.
>
> Not really. You see, this is actually a model of trust in action. Some
> of the societies that peer review scientific work have the highest
> reputation, others do not. While there's the occasional slip-up this
> in general works very well. (Much of it, of course, hinges on the
> ability of the readers of said peer reviewed work to be able to check
> the references...) The IEEE stands behind the printed copy, and they
> are able to do so with some authority, and confidence.

I supposed either you misunderstood me or vice versa. What I mean is this: Suppose there is circulated on the internet a document (of whatever type) bearing a name X. Suppose X is unique. How do we know that this document is really from the person having the name X? (This has nothing to do peer review or such stuffs.)

M. K. Shen

Subject: Re: Memo to the Amateur Cipher Designer
Date: 17 Nov 1998 10:11:58 -0500
From: juola@mathcs.duq.edu (Patrick Juola)
Message-ID: <72s3numhu1@quine.mathcs.duq.edu>
References: <36516E2F.FB265F4A@stud.uni-muenchen.de>
Newsgroups: sci.crypt
Lines: 58

In article <36516E2F.FB265F4A@stud.uni-muenchen.de>,
Mok-Kong Shen <mok-kong.shen@stud.uni-muenchen.de> wrote:
>Stefan Axelsson wrote:
>>
>> In article <36512E2C.CB9FB39B@stud.uni-muenchen.de>,
>> Mok-Kong Shen <mok-kong.shen@stud.uni-muenchen.de> wrote:
>>
>> >I see rather the same problems with the originals of published
>> >Web pages. Due to the inherent insecurity of the internet (see faked
>> >mail addresses, for example) there is the question of authentication
>> >of the author. Of course, paper publications actually also have
>> >the same problem.
>>
>> Not really. You see, this is actually a model of trust in action. Some

>> of the societies that peer review scientific work have the highest
>> reputation, others do not. While there's the occasional slip-up this
>> in general works very well. (Much of it, of course, hinges on the
>> ability of the readers of said peer reviewed work to be able to check
>> the references...) The IEEE stands behind the printed copy, and they
>> are able to do so with some authority, and confidence.
>
>I supposed either you misunderstood me or vice versa. What I mean
>is this: Suppose there is circulated on the internet a document (of
>whatever type) bearing a name X. Suppose X is unique. How do we
>know that this document is really from the person having the
>name X? (This has nothing to do peer review or such stuffs.)

Actually, it does have something to do with peer review. The IEEE (for example) has implicitly "signed" or "authenticated" the claims made in its published work. So if you have an authentic copy of an IEEE publication, it implicitly stands behind the authenticity and accuracy of the works contained between the covers. And, of course, the publication is a physical copy and as such is relatively tamper-proof; it's easy to tell if someone has (naively) ripped out several pages and replaced them with something else.

Of course, it isn't perfect -- I rarely get my journal articles straight from the IEEE, and it's possible that the CIA or someone could break into my office in the middle of the night and replace my journal copies with indetectable forgeries -- or that they could conspire with my *librarian* with the same intention but much worse result.

So, in direct answer to your question, if there is a document on the Web purporting to be from me, that means little or nothing. If you got it from a Web site demonstrably owned by me, that means significantly more. *OR* if you got it from a Web site whose judgement and trustworthiness you accept, that also means more.

In the case of physical objects, if you don't believe that I have the skill to indetectably tamper with a physical journal, you can also trust the copy of IEEE Trans. Info. Thy. that I lent you from my shelves.

-kitten

Subject: Re: Memo to the Amateur Cipher Designer
Date: 23 Nov 1998 10:18:23 GMT
From: sax@rmovt.rply.ce.chalmers.se (Stefan Axelsson)
Message-ID: <73bcpf$nk8$1@nyheter.chalmers.se>
References: <72s3numhu1@quine.mathcs.duq.edu>
Newsgroups: sci.crypt
Lines: 21

In article <72s3numhu1@quine.mathcs.duq.edu>,
Patrick Juola <juola@mathcs.duq.edu> wrote:

>Actually, it does have something to do with peer review. The
>IEEE (for example) has implicitly "signed" or "authenticated"
>the claims made in its published work.

My point exactly, and that's even side stepping the issue of the review of the content, the most valuable service performed by the IEEE in this case.

And of course, the beauty of there being several hard copies made of each publication, makes it trivial for the reader to get his material from several sources, should he lack trust in any single one of them. The converse is of course not true of the web.

Stefan,
--
Stefan Axelsson Chalmers University of Technology
sax@rmovt.rply.ce.chalmers.se Dept. of Computer Engineering
(Remove "rmovt.rply" to send mail.)

Subject: Re: Memo to the Amateur Cipher Designer
Date: Tue, 17 Nov 1998 14:08:18 -0600
From: jgfunj@EnqvbSerrGrknf.pbz (W T Shaw)
Message-ID: <jgfunj-1711981408180001@dialup176.itexas.net>
References: <36516E2F.FB265F4A@stud.uni-muenchen.de>
Newsgroups: sci.crypt
Lines: 26

In article <36516E2F.FB265F4A@stud.uni-muenchen.de>, Mok-Kong Shen
<mok-kong.shen@stud.uni-muenchen.de> wrote:
>
> I supposed either you misunderstood me or vice versa. What I mean
> is this: Suppose there is circulated on the internet a document (of
> whatever type) bearing a name X. Suppose X is unique. How do we
> know that this document is really from the person having the
> name X? (This has nothing to do peer review or such stuffs.)
>

You could mail or email a question to the author. It should be meaningful to post an email address, real address, and phone number. This either checks to the author, or it does not. Perhaps we need some bind PO/call forwarding schemes that can verify the identity of their subscribers if the direct approach is not acceptable.

Sending a letter via PO without giving a return address is a fairly good way to get an anonymous message across. It is easy to recognize that threats and harassment via snail mail are a federal offense, and you would be bound to respect a directive to not send something again to a party telling you not to. The big problem is how to transfer this situation to the net and neither lose nor add anything.
--

Your future is ahead of you.--Thomas E. Dewey

Decrypt with ROT13 to get correct email address.

Subject: Re: Memo to the Amateur Cipher Designer
Date: 17 Nov 1998 14:12:50 -0500
From: juola@mathcs.duq.edu (Patrick Juola)
Message-ID: <72shrinae1@quine.mathcs.duq.edu>
References: <jgfunj-1711981408180001@dialup176.itexas.net>
Newsgroups: sci.crypt
Lines: 30

In article <jgfunj-1711981408180001@dialup176.itexas.net>,
W T Shaw <jgfunj@EnqvbSerrGrknf.pbz> wrote:
>In article <36516E2F.FB265F4A@stud.uni-muenchen.de>, Mok-Kong Shen
><mok-kong.shen@stud.uni-muenchen.de> wrote:
>>
>> I supposed either you misunderstood me or vice versa. What I mean
>> is this: Suppose there is circulated on the internet a document (of
>> whatever type) bearing a name X. Suppose X is unique. How do we
>> know that this document is really from the person having the
>> name X? (This has nothing to do peer review or such stuffs.)
>>
>You could mail or email a question to the author. It should be meaningful
>to post an email address, real address, and phone number.

Not especially helpful, I'm afraid. These things are *SO* ephemeral
that they can't be reliably checked after six months or so. And
even if you send mail and get a response back, how do you know that
you're getting the right person?

I could, for instance, easily create an Email address :

monica.lewinsky@quine.mathcs.duq.edu

and circulate a document claiming to "tell all" (not that there's
much left to tell). When I receive mail to that address, I could
easily forge something purporting to confirm that Monica *does*
dial into an obscure machine at an obscure Pittsburgh university.
And if you believe *that*, you'll probably believe anything.

-kitten

Subject: Re: Memo to the Amateur Cipher Designer
Date: Sun, 22 Nov 1998 12:47:22 -0600
From: jgfunj@EnqvbSerrGrknf.pbz (W T Shaw)
Message-ID: <jgfunj-2211981247370001@207.101.116.115>
References: <72shrinae1@quine.mathcs.duq.edu>
Newsgroups: sci.crypt
Lines: 29

In article <72shrinae1@quine.mathcs.duq.edu>, juola@mathcs.duq.edu
(Patrick Juola) wrote:
> >>
> >You could mail or email a question to the author. It should be meaningful
> >to post an email address, real address, and phone number.
>
> Not especially helpful, I'm afraid. These things are *SO* ephemeral
> that they can't be reliably checked after six months or so. And
> even if you send mail and get a response back, how do you know that
> you're getting the right person?
>
The problem of authentication is big. You could go on content
alone.....so anything that I have said that was incorrect, I can claim was
a forgery. There is something said for meeting people physically,
creating a history on which verification can be based.

Now, if someone showed up and claimed to be David Sternlight, who would
believe it, and who could verify it? We could test his rhetoric against
the wealth of unique logic patterns claimed to have originated by him.
Such a scientific test, a sort of linguistic fingerprint, might surfice.

It would be hard to orallly identify a certain affected attribute of my
writings without me doing a Victor Borge style punctuated presentation;
haven't you noticed the gimmick?
--

Jail would not be a cheerful place for revisiting ones recollections.

Decrypt with ROT13 to get correct email address.

Subject: Re: Memo to the Amateur Cipher Designer
Date: Wed, 11 Nov 1998 16:55:42 GMT
From: schneier@counterpane.com (Bruce Schneier)
Message-ID: <364bc0f8.4701632@news.visi.com>
References: <36494616.1500C031@stud.uni-muenchen.de>
Newsgroups: sci.crypt
Lines: 29

On Wed, 11 Nov 1998 09:08:54 +0100, Mok-Kong Shen
<mok-kong.shen@stud.uni-muenchen.de> wrote:
>I am taking this opportunity to say a bit more about peer review
>of patent publications. A scientific paper is reviewed by a number
>of peers before publishing. Afterwards, there is public review.
>A reader may object by writing a 'letter to editor'. For a patent
>(in the case of US, not other countries!) there is no review by
>peers before patent publication. But afterwards the document is
>examined by the professionals in the patent divisions of a number
>of firms whose own patents may be concerned. If there are objections
>there will be a legal issue in court. (This is at least true in
>countries other than US.)

I don't know what company you work for, but I am willing to concede
that your company acts in the manner you describe. No company I have
ever worked with has behaved in this manner, and I know of no company
that does so. Almost all patents are examined by almost nobody. And
I assure you that if a company sees a patent and has objections, in
the great majority of cases there is no legal issue in court. Court
is expensive; most companies have better things to do with their time.

But again, I have no direct experience with the company (or companies)
that you work for.

Bruce
****
Bruce Schneier, President, Counterpane Systems Phone: 612-823-1098
101 E Minnehaha Parkway, Minneapolis, MN 55419 Fax: 612-823-1590
Free crypto newsletter. See: http://www.counterpane.com

Subject: Re: Memo to the Amateur Cipher Designer
Date: Wed, 11 Nov 1998 19:53:11 +0100
From: Mok-Kong Shen <mok-kong.shen@stud.uni-muenchen.de>
Message-ID: <3649DD17.867829C@stud.uni-muenchen.de>
References: <364bc0f8.4701632@news.visi.com>
Newsgroups: sci.crypt
Lines: 25

Bruce Schneier wrote:

> I don't know what company you work for, but I am willing to concede
> that your company acts in the manner you describe. No company I have
> ever worked with has behaved in this manner, and I know of no company
> that does so. Almost all patents are examined by almost nobody. And
> I assure you that if a company sees a patent and has objections, in
> the great majority of cases there is no legal issue in court. Court
> is expensive; most companies have better things to do with their time.
>
> But again, I have no direct experience with the company (or companies)
> that you work for.

From which part of my writing did you infer that I argued in
reference to a or the company I work for??? I argued in general
terms, didn't I? As I wrote previously I know that many organic
chemical compounds, for example, are patented. A competitor can't
use these or have to pay license fees. That's why large chemical
firms need people knowledgeable in such patents in order that
they can do their business properly. These companies have so much
money that the court expenses are really entirely negligible. This
is one example that I happen to be able to present. Are you going
to counter with sentence like 'But in crypto it is different'?

M. K. Shen

Subject: Re: Memo to the Amateur Cipher Designer
Date: Wed, 11 Nov 1998 22:33:11 GMT
From: schneier@counterpane.com (Bruce Schneier)
Message-ID: <364b1060.25032108@news.visi.com>
References: <3649DD17.867829C@stud.uni-muenchen.de>
Newsgroups: sci.crypt
Lines: 41

On Wed, 11 Nov 1998 19:53:11 +0100, Mok-Kong Shen
<mok-kong.shen@stud.uni-muenchen.de> wrote:

>Bruce Schneier wrote:
>
>> I don't know what company you work for, but I am willing to concede
>> that your company acts in the manner you describe. No company I have

>> ever worked with has behaved in this manner, and I know of no company
>> that does so. Almost all patents are examined by almost nobody. And
>> I assure you that if a company sees a patent and has objections, in
>> the great majority of cases there is no legal issue in court. Court
>> is expensive; most companies have better things to do with their time.
>>
>> But again, I have no direct experience with the company (or companies)
>> that you work for.
>
>From which part of my writing did you infer that I argued in
>reference to a or the company I work for??? I argued in general
>terms, didn't I?

Don't know. I was just giving you the benefit of the doubt that your
experiences may have differed from my own. As you show below, you
have experience (or at least knowledge) from chemical firms, which is
knowledge that I lack.

>As I wrote previously I know that many organic
>chemical compounds, for example, are patented. A competitor can't
>use these or have to pay license fees. That's why large chemical
>firms need people knowledgeable in such patents in order that
>they can do their business properly. These companies have so much
>money that the court expenses are really entirely negligible. This
>is one example that I happen to be able to present. Are you going
>to counter with sentence like 'But in crypto it is different'?

Nah. I'm going to drop the thread.

Bruce
****
Bruce Schneier, President, Counterpane Systems Phone: 612-823-1098
101 E Minnehaha Parkway, Minneapolis, MN 55419 Fax: 612-823-1590
Free crypto newsletter. See: http://www.counterpane.com

Subject: Re: Memo to the Amateur Cipher Designer
Date: Thu, 12 Nov 1998 10:25:16 +0100
From: Mok-Kong Shen <mok-kong.shen@stud.uni-muenchen.de>
Message-ID: <364AA97C.5EDD0267@stud.uni-muenchen.de>
References: <364b1060.25032108@news.visi.com>
Newsgroups: sci.crypt
Lines: 39

Bruce Schneier wrote:

> >From which part of my writing did you infer that I argued in
> >reference to a or the company I work for??? I argued in general
> >terms, didn't I?
>
> Don't know. I was just giving you the benefit of the doubt that your
> experiences may have differed from my own. As you show below, you
> have experience (or at least knowledge) from chemical firms, which is
> knowledge that I lack.

But this is simply example of so-called 'common knowledge'. Some
one happens to know this, another happens to know that. Maybe
you know more about space travel than I, while I know a bit more
about gene manipulations than you. That patents are important in
the 'practice' (as against pure theory) of a large number of
professions should be well-known. All of us use, for example, laser
printers. But a large number of fonts are protected (the fonts of
Knuth are free). We users of the printers don't need to know that
but the 'professionals' in the industry concerned with printers
have to know the details.

Let me say some more words against the be-littlement of patent
publications. I'll choose an analogy which I already have used.
DES is described in your well-known book. But where did you get
the informations? Maybe you got that from another author. But
then where did he get that? Ultimately one comes to the original
government publication, if one continues asking. So if that original
document (which is perhaps not a scientific paper in the eyes of
certain academics) doesn't get cited very often, that doesn't mean
anything, in particular it does not affect the scientific value and
significance (contribution) of that document. If a patented crypto
is really good, it will be popular and more people will study it,
eventually publishing papers on it. Whether the authors of the
papers cite the patent document is in my view not very essential.
One thing is on the other hand certain, namely that without the
publishing of the original document these papers could not exist.

M. K. Shen

Subject: Re: Memo to the Amateur Cipher Designer
Date: Fri, 13 Nov 1998 10:30:55 +0100
From: Mok-Kong Shen <mok-kong.shen@stud.uni-muenchen.de>
Message-ID: <364BFC4F.4AED9332@stud.uni-muenchen.de>
References: <364b1060.25032108@news.visi.com>
Newsgroups: sci.crypt
Lines: 46

To my post of 12 Nov 1998 10:25:16 +0100 I like to add the following

information related to patents which I have just (an hour before) acquired. This consists of the two URLs:

http://www.wired.com/news/print_version/technology/story/16180.html?wnpg=all

http://jya.com/rivest111098.htm

The first is an article 'Patent May Threaten E-Privacy' which reported that much concern has been directed to a pending patent of P3P (Platform for Privacy Preferences) that could have significant impact for the internet. This shows that people are vigilent even on pending patents which are not officially published in the US (different from other countries).

The second has the title 'US5835600 Block encryption algorithm with data dependent rotations' and is a patent issued to Prof. R. Rivest. This is interesting in that only three days after the issue of the patent it is already to be found on a Web page maintained by some private person, (highly probably) in contradiction to the thesis that 'Almost all patents are examined by almost nobody'.

I think that it is universally true that where big money and/or its equivalents, e.g. personal survival, are involved there will be proportionately high attention paid by the community. Otherwise it could well happen under circumstances that very few people read a very high quality scientific paper that has no practical relevances. This is lamentable but is a fact of life.

BTW, although I haven't yet closely studied Rivest's patent, I guess that there is a certain (maybe only weak) parallel of a small part of his idea with a small part of the idea underlying my WEAK3-E, since I also make use of rotations in block encryption (that is controled by a hash value which is dependent on the plain text being processed and hence data dependent (though differing in the detailed manner from the patent)).

M. K. Shen

M. K. Shen, Postfach 340238, D-80099 Muenchen, Germany
+49 (89) 831939 (6:00 GMT)
mok-kong.shen@stud.uni-muenchen.de
http://www.stud.uni-muenchen.de/~mok-kong.shen/ (Last updated: 10th October 1998. origin site of WEAK1, WEAK2, WEAK3 and WEAK3-E. Containing 2 mathematical problems with rewards totalling US$500.)

Subject: Re: Memo to the Amateur Cipher Designer
Date: Wed, 18 Nov 1998 14:21:39 +0100
From: Mok-Kong Shen <mok-kong.shen@stud.uni-muenchen.de>
Message-ID: <3652C9E3.1B972124@stud.uni-muenchen.de>
References: <364b1060.25032108@news.visi.com>
Newsgroups: sci.crypt
Lines: 818

There has been recently some one-to-one discussions on themes occuring in this thread between Bruce Schneier and me, i.e. discussions outside of the group. I am posting in the following three e-mails (separated from one another by triple lines of *****) :

(1) Shen to Schneier 17th Nov

(2) Schneier to Shen 13th Nov

(3) Shen to Schneier 16th Nov

The motivation of posting these is given in (1). (I chose to post (2) instead of a previous e-mail of mine since that material is entirely contained in (2)).

I have tried my best to argue, particularly in (3), especially for the following:

a. Patents play in science and technology a significant role which cannot be and is not ignored by the academic community (which is a subset of the scientific/professional community).

b. Sci.crypt is not a chatroom.

Concerning the special field of cryptology I like to stress that I am not arguing that patent documents are excellent sources for one to search for good crytos but I am definitely against the notion that they are all valueless to science.

I hope that there will be further discussions in the group to make very clear what is true and what is not.

M. K. Shen

****
****
****

I had today the opportunity to talk to a number of persons interested in patents and mentioned the topic vehemently disputed by us in the current time point. They found that our one-to-one discussion contain very interesting informations and opinions that should be accessible to a larger circle of readers so as to induce more discussions on the position of patents in scientific research, resulting in better insight about this essential theme. I therefore suggest that you delay answering my mail of yesterday by some 24 hours. I'll try to post my previous two responses tomorrow afternoon as a base upon which we and others of the group could carry on further discussions.

Cheers,

M. K. Shen

****
****
****

At 07:29 PM 11/13/98 +0100, Mok-Kong Shen wrote:
>Bruce Schneier wrote:
>
>> I'm still not sure what you're position is. Are you mad at me because I am
>> describing the academic position at patents? Do you believe I am wrong
>> when I say that patents are not cited very often? Do you think that
>> academics do read patents, and just don't cite them for some nefarious
>> reasons? Have you still note been convinced that patents do not go through
>> the same peer review process that academic papers do? Do you believe
>> that patent applications do count as publications for tenure?
>
>Please don't feel offended in any way in the following. I attempt
>to express clearly and directly my thoughts without spending too
>much time in the choice of wordings to make them 'look' nicer.
>
>I am not 'mad' at you at all. I feel (subjectively) I am arguing
>within the framework of scientific discussions, arguing as strong as
>possible for the truth (as believed by me, before I am shown to
>be wrong). In other words, I have no 'personal' feeling against you.
>(Why shoud I have one? You have at no time hurt me!) You described
>the academic position at patents. I doubt that this is so.

Do you doubt that my description is so, or that I am decribing it?

>I don't
>like to argue too much about that. However it appears to me that you
>are defending the position of the academics.

I am no defending. I am describing. As you have pointed out, I am one
of the few who regularly cites patents.

>It is my basic view that
>the academics should not neglect the informations in the patent
>publications because there are (at least some) valuable scientific
>informations there.

Fine. That is a perfectly valid position. Please take it up with
people
who believe the reverse position.

>Noting that we are (at least partly) discussing
>on the general level, i.e. not restricting exclusively to crypto
>matters, I believe that some academics do read patents.

Of course some do. Most do not.
'
>Just two
>supporting arguments: 1) Some patents are applied by academics.

In cryptography at least, most academics hate patents. You wouldn't
believe the bile that such a discussion brings up at a crypto
conference.

>2) In engineering, for example, there are patented machine parts,
>patented production methods, etc. These are to be taught to the
>students. So the professors have to be acquainted with them.

No. It is possible to teach X without reading the patent on X, just as
it is possible to teach RSA without reading the RSA patent. Actually,
it's better not to. The description is terrible and misleading. The
academic paper is much easier to read, and much more useful as
a teaching tool.

>I agreed
>that patents are not cited very often and have (twice) given my
>explanation of the phenomenon. Academics don't cite patents because
>of convenience.

I don't understand this. It is no less convenient to cite one thing or
another,
assuming you've read them both. Academics don't cite patents because

they don't read them.

>This is understandable (if something is well-known,
>it is pedantic to cite the original document everytime one writes
>about it).

The original document cited is the paper, not the patent. Look at RSA, DES, Chaum's blind signatures stuff, and pretty much everything else in cryptography that has been patented. Find me one paper where the IDEA patent is cited instead of the IDEA paper. Or one where the RC5 patent is cited instead of the RC5 paper. I try to cite patents, because
I don't want academics to be able to ignore the problems of their colleagues patenting their reserch. But I am a minor minority.

>I have argued recently (though only implicitly expressed)
>that it is the 'total' review (not the review before publication) that
>should be taken into consideration. A paper receives proportinally
>more review before publication than a patent (the US case is singular
>but can be subsumed here) but if the content is worth scientifically
>there is no inherent reason why the amount of total (before and
>after publication) review (review by the scientific community, not
>solely by the academics!) should be less in one case than the others.

A patent receives no peer review. From the point of view of an academic, a patent receives no review. In the past, I have tried explaining the patent proceess and explaining the referee process for a paper. I have suggested that you get other opinions if you do not believe me. You can do whatever you want, though, and believe whatever you want.

>For a paper there are a number of referees who because of their faith
>to science conduct a rigorous examination. For a patent there are the
>people of the patent divisions of the competitors, who are
>professionals in the fields and are instructed by their employers to
>conduct a rigorous examination because revenues could be at stake.

The above does not happen, in general. I'm sorry if you don't believe me.
I assumed you believed the above based on person experience, but you got annoyed when I suggested that. I have no idea what to do now, and you are welcome to believe whatever you want.

>I am of the opinion there is a rough equivalence in this respect,
>even if the motivations of the examination differ.

This is not true. Again, you can believe whatever you want. Please find
others to discuss this with; possibly multiple opinions will convince you.
(Please do not take the above as a suggestion that you spend money seeking professional advice, which would not be right in me suggesting.)

>Patent applications
>might not count for the tenure. I don't know. (But these do count for
>those academics that have applied for patents!)

Patent applications do not count as publications, period. They do not count as publications when tenure is discussed. The ability to get patents may be a goodness for a university in and of itself, though.

>However, it is my
>view that this situation (if it is indeed true) is not correct and
>should get changed. (I can only hope but can offer of course no
>means of effecting such change.)

Good luck.

>I hope that the above answers the
>bunch of questions you posed above. Please let me know if some points
>are not yet adequately covered.

They do. I understand your position. I just have no further interest in
debating it.

>> But as I said in Usenet, I really don't want to carry on this conversation.
>> It
>> is not fun. You are not listening, either to me or to others on the
>> newsgroup.
>> I feel like you are blaming me personally simply because you don't like
>> what I am saying. If this kind of thing happens socially, I generally walt
>> away from the offending person. I did so, and you have followed me and
>> sent me personal mail. So I am talking again. If I still find the
>> conversation
>> unpleasant, I will walk away again.
>
>It is my humble opinion that people engage in scientific discussions
>because they desire to find the truth and not becuase they desire
>to find fun and pleasure.

That is the difference. You are engaging in a scientific discussion. I am
merely chatting on Usenet. We are invested in differing amounts.

>In order to find the truth one is ready to
>pay the price, if necessary, of unpleasantness. I am excluding
>impoliteness here which I hate and which unfortunately I experienced
>often in discussion groups. I can assure you that from my standpoint
>I have at no time point blamed you (in the sense of the word 'blame'
>as I understand it). But it is true that I am very hard-necked (is
>this a correct English word?) in scientific discussions.

"Stiff necked" is proper. But nice choice.

>If something
>is in my opinion wrong, I alway say it 'very' 'very' directly (without
>'speaking through the flowers'). Perhaps you are not used to
>discussion partners of my kind.

I am.

>After all, to take part in a scientific
>discussion is free will, there is no obligation. If one gets tired,
>feel the stuff uninteresting, or for whatever reason, one is
>entirely free to stop arguing. You are not bound to reply to posts
>of anybody. Due to my hard-neckedness I used to continue discussion,
>however, up to the very end. (Not very long ago I was engaged in a
>discussion on possible parapsychological influences on phsical events
>and in a discussion on Rivest's chaffing and winnowing. Both cases had
>cost me quite a lot of extra time and energy because a few discussion
>partners were not arguing scientifically in my humble view.)

That's fine. I respect this.

>> >The second has the title 'US5835600 Block encryption algorithm with
>> >data dependent rotations' and is a patent issued to Prof. R. Rivest.
>> >This is interesting in that only three days after the issue of
>> >the patent it is already to be found on a Web page maintained by some
>> >private person, (highly probably) in contradiction to the thesis
>> >that 'Almost all patents are examined by almost nobody'.
>>
>> I knew about it the day it was released. This was important. Again, I
don't
>> see how it relates to a discussion as whether or not patents are generally
>> read by academics. They are not. My apologies if you don't like this
>> fact.
>
>You misunderstood me. My point here is not concerned with the academics.
>The point is that patents do get attention from the scientific
>(professional) community (which is not identical to the set of
>academics!).

Of course they get attention from the scientific community; I didn't
think
that
was an issue. If I misquoted you, it is because I could think of no
other
reason
for you to bring the item up. If you simply brought it up to make the
point
that patents get attention from professionals, then I agree with you.

I do not believe that "get attention from" and "are worth an academic's
time
to read" are very different things.

>> >I think that it is universally true that where big money and/or its
>> >equivalents, e.g. personal survival, are involved there will be
>> >proportionately high attention paid by the community. Otherwise
>> >it could well happen under circumstances that very few people read
>> >a very high quality scientific paper that has no practical relevances.
>> >This is lamentable but is a fact of life.
>>
>> Yes, and the academic community is generally unconcerned with patents.
>> The examples above are from the business community.
>
>You seem to stick to the issue of academic community. I am concerned
>with the scientific (professional) community, including the
>scientists that work for the business firms. Much of the divergence
>of our discussions can be traced to this fact. Since most of
>the readers of the group are not academics, not to say having tenures,
>I believe that my broader standpoint is the more appropriate one
>for the present discussion than yours.

Oh. I didn't think this was about the professional community. I
thought we
were talking about patents being 1) worthy of academic citation, 2)
worthy
of "publication" status on par with real publications, and 3) the
recipient
of peer review similar to real publications.

>> >BTW, although I haven't yet closely studied Rivest's patent, I guess
>> >that there is a certain (maybe only weak) parallel of a small part
>> >of his idea with a small part of the idea underlying my WEAK3-E, since
>> >I also make use of rotations in block encryption (that is controled
>> >by a hash value which is dependent on the plain text being processed
>> >and hence data dependent (though differing in the detailed manner
>> >from the patent)).

>>
>> And there's an algorithm in my book that has data dependent rotations.
>> And IBM makes a claim about them.
>>
>> Again, I don't see what this has to do with the fact that patents are not
>> generally cited as academic papers, are not generally considered
publications
>> by academics, and are not subjected to the same peer review process
>> as academic papers.
>
>You misunderstood me. Actually here it is my fault. I wanted to put
>before the string 'BTW' above the phrase 'Something off-topic:'.
>But an uncontrolled movement of my hand caused the message to be
>sent without that phrase. My intention here is more 'personal'
>(or egoistic), hoping that someone of the group would say something
>about the idea of using rotations (which I also used in my algorithm
>and of which I like to hear some opinions for eventually improving
>my own design).

Oh.

>I hoped that I have given you a sufficiently understandable response.
>If you have further questions or points, I shall be very glad to
>answer them and discuss with you. As I said, I am very hard-necked in
>scientific discussions (not so in private social discussions) and
>I express my thougts plainly without 'artificial' modifications.
>I hope I have not said anything impolite, using bad words etc.,
>since I have not a single reason to be angry with you. Maybe
>some words were 'strong' because I don't like to 'speak through
>the flowers' but I am not conscious of having ever said anything that
>does not correspond to plain facts. Please feel free to point out
>directly, quoting my writing, if this is not true in your opinion,
>so that I may learn something in this respect.

Cheers,
Bruce
****
Bruce Schneier, President, Counterpane Systems Phone: 612-823-1098
101 E Minnehaha Parkway, Minneapolis, MN 55419 Fax: 612-823-1590
Free crypto newsletter. See: http://www.counterpane.com

****
****
****

Bruce Schneier wrote:>
> At 07:29 PM 11/13/98 +0100, Mok-Kong Shen wrote:

> >(Why shoud I have one? You have at no time hurt me!) You described
> >the academic position at patents. I doubt that this is so.
>
> Do you doubt that my description is so, or that I am decribing it?

I don't doubt at all the sincerity with which you gave your description.
I simply doubt that what you described really corresponds to real world
facts. (This lies at the foundation of our discussions.)

> >I don't
> >like to argue too much about that. However it appears to me that you
> >are defending the position of the academics.
>
> I am no defending. I am describing. As you have pointed out, I am
> one of the few who regularly cites patents.

Once again, could we just forget about the academics in our discussions
and concentrate on the scientific (professional) community as a whole
(and even including the amateurs)? Note that by taking this broader
scope the academics are NOT excluded, they become simply a minority.

> >It is my basic view that the academics should not neglect the
> >informations in the patent publications because there are (at
> >least some) valuable scientific informations there.
>
> Fine. That is a perfectly valid position. Please take it up with
> people who believe the reverse position.
>
> >Noting that we are (at least partly) discussing
> >on the general level, i.e. not restricting exclusively to crypto
> >matters, I believe that some academics do read patents.
>
> Of course some do. Most do not.

Just consider those prominent persons as Rivest, Shamir, Schnoor
and Chaum. (As far as I know Chaum is not a professor but he is
working for a scientfic institution (comparable to an academy of
science) in Netherland.) These all have patents. They must read
other patents (not merely their own) when applying for patents.
If some professors do not 'directly' read patent publications they
'indirectly' read them (see below). Does that make an essential
difference in the context of the present discussion? (If you don't

know Russian and you read English translations of Tolstoi, does that mean Tolstoi's original work is valueless??)

I have two friends who are professors and who have some patens. One of them asked me recently even a question on convenient accesses to patent materials. (It happened that I could help him a bit because both the German and the European patent offices are in Munich.) The other made essential contributions to energy dissipation devices to prevent destructive vibrations of structures due to wind forces. He teaches his (patented) research stuffs to students and gives also lectures on that to professional engineers.

By PURE chance, it happens that TODAY (16th Nov.) begins an information week organized by LMU, one of the universities in Munich, on patents for the students, in order to raise their awareness to the importance of patents to technology transfer. You can look at their program at:

http://www.uni-muenchen.de/kft/patente/infowoche.html

Even if you don't read German you can verify that e.g. it includes visits to the two patent offices in Munich, etc. Note that LMU is more humanities oriented, almost all technological faculties are concentrated in another university, the TU (Technical University), which by nature is more involved with patents.

> >Just two
> >supporting arguments: 1) Some patents are applied by academics.
>
> In cryptography at least, most academics hate patents. You wouldn't
> believe the bile that such a discussion brings up at a crypto conference.
>
> >2) In engineering, for example, there are patented machine parts,
> >patented production methods, etc. These are to be taught to the
> >students. So the professors have to be acquainted with them.
>
> No. It is possible to teach X without reading the patent on X, just as
> it is possible to teach RSA without reading the RSA patent. Actually,
> it's better not to. The description is terrible and misleading. The
> academic paper is much easier to read, and much more useful as
> a teaching tool.

Well, what matters if a professor does not read the patent document but his fellow professor reads that and describes it in detail in a text book or paper so that the first professor can teach the stuff to his students? (Isn't it even better, since in putting the stuff in a text book he has to well digest the meterial in the patent publication and thus is at the same time doing sort of 'peer review'?) What matters if you, when you wrote on DES in your book, did not consult the original NBS document but get all materials, say, from the book of H. Kazan who makes the material more palatable to the readers by providing an implemenation with example results?? Does that mean the NBS document is unimportant, to be ignored?? The NBS document is, on the contrary, the 'bible' for implementing DES!

> >I agreed
> >that patents are not cited very often and have (twice) given my
> >explanation of the phenomenon. Academics don't cite patents because
> >of convenience.
>
> I don't understand this. It is no less convenient to cite one thing or
> another,
> assuming you've read them both. Academics don't cite patents because
> they don't read them.

Covered sufficiently above, I think. Why are you attaching so much weight to the 'academics' in the present discussions? Excuse me for asking you a more personal question. Are you teaching in a university? Assuming the answer is 'no', are you doing 'peer review' (a term according to your definition) when you do work in the program committe of FSE?

> >This is understandable (if something is well-known,
> >it is pedantic to cite the original document everytime one writes
> >about it).
>
> The original document cited is the paper, not the patent. Look at RSA,
> DES, Chaum's blind signatures stuff, and pretty much everything else
> in cryptography that has been patented. Find me one paper where the
> IDEA patent is cited instead of the IDEA paper. Or one where the RC5
> patent is cited instead of the RC5 paper. I try to cite patents, because
> I don't want academics to be able to ignore the problems of their
> colleagues patenting their reserch. But I am a minor minority.

At least in Germany, stuffs already published are not viable for patent considerations. That's why often results are at first kept secret. I once heard a public lecture by a professor on some electronic stuffs. He declined to give details of the work of one of his projects, saying that even a revelation in a talk (not publication in journal) could endanger his planned patent application (for that would be equivalent to publication for the patent office). Now if after the issue of a patent there are papers or books describing sufficiently well the stuff in the patent document, then one certainly perfers

to cite the papers or books, since they are more easily accessible.
(This situation has changed. The German patents are e.g. on-line!)
But why does this fact render the value of the patent publications to zero?? Could there be papers in such cases at all WITHOUT the patent publications?

Let me put one question? Should we properly argue on patents (the 'invention' which happens to be published by the government) OR should we argue extremely over-proportionately on the physical sheets of paper on which some words and sentences and diagrams of the inventor are put? We began the present patent argument when I said that Terry Ritter's algorithms have a special (singular) position among the amateur ciphers because there are patents on them . Suppose one of his patents is indeed of comparable quality to IDEA, what is going to happen? For IDEA the history shows that there will be papers that are more easily accessible to the public. On the assumption just made, why should there not be papers for Ritter's designs as well?? You could only argue the other way round, namely that because some long time has elapsed without seeing papers on Ritter's designs, his work appears to be probably not especially good. But if you argue that, you implicitly assume that there are quite a number of people who have already examined his stuffs. (Note that from the very beginning, IDEA is a patent and Ritter's design is a patent. Why should one give from the outset less attention to one patent document than the other??)

> >I have argued recently (though only implicitly expressed)
> >that it is the 'total' review (not the review before publication) that
> >should be taken into consideration. A paper receives proportinally
> >more review before publication than a patent (the US case is singular
> >but can be subsumed here) but if the content is worth scientifically
> >there is no inherent reason why the amount of total (before and
> >after publication) review (review by the scientific community, not
> >solely by the academics!) should be less in one case than the others.
>
> A patent receives no peer review. From the point of view of an
> academic, a patent receives no review. In the past, I have tried
> explaining the patent proceess and explaining the referee process for
> a paper. I have suggested that you get other opinions if you do not
> believe me. You can do whatever you want, though, and believe
> whatever you want.

I am on the thread. If there are opinions I'll see them and argue!
Only region is to be 'believed'. Science has to be established on facts.
Please pay attention to my repeated appeal to concentrate on the scientific community as a whole and NOT to consider only the academics!
I said more than once that particularly the professionals in the patent divisions of commercial/industrial firms are ALSO 'peers'.
I happen to have cast a glance into the recent issue of New Scientist (7th Nov.). There is a job offer on p.75 to a 'scientist with up-to-date biotechnology insight to manage and protect the intellectual property and patent interest of the company'. Doesn't that say something to you in the present context?

> >For a paper there are a number of referees who because of their faith
> >to science conduct a rigorous examination. For a patent there are the
> >people of the patent divisions of the competitors, who are
> >professionals in the fields and are instructed by their employers to
> >conduct a rigorous examination because revenues could be at stake.
>
> The above does not happen, in general. I'm sorry if you don't believe me.
> I assumed you believed the above based on person experience, but you
> got annoyed when I suggested that. I have no idea what to do now, and
> you are welcome to believe whatever you want.

If there are not often legal issues about patents, it is because the examiners have done a good job and because there are (in countries other than US) public review before patents are issued.

Just the other day I read something about patents of electronics in handys. There are competing patents of Philips and Hitachi with different merits. Don't tell me that there are no professionals who critically compare and evaluate these.

> >I am of the opinion there is a rough equivalence in this respect,
> >even if the motivations of the examination differ.
>
> This is not true. Again, you can believe whatever you want. Please find
> others to discuss this with; possibly multiple opinions will convince you.
> (Please do not take the above as a suggestion that you spend money
> seeking professional advice, which would not be right in me suggesting.)

This IS happening on the thread which is still active. We are conducting here a one to one private conversation. Do you prefer that I post this mail and the last response of mine to the group in order to elicite more discussions? I'll do it if you think that's better.

> >Patent applications
> >might not count for the tenure. I don't know. (But these do count for
> >those academics that have applied for patents!)
>
> Patent applications do not count as publications, period. They do not

> count as publications when tenure is discussed. The ability to get
> patents may be a goodness for a university in and of itself, though.

So government publications are NO publications according to you?
Even a newspaper IS a publication! What dictionary do you use?
How many persons in the world have tenures? How many professionals
are there? Should a handful of guys (here the academics) choose to
call white black, all the rest of the mankind could FORGET them!!!
As I said nowadays science is NO LONGER religion!

> >However, it is my
> >view that this situation (if it is indeed true) is not correct and
> >should get changed. (I can only hope but can offer of course no
> >means of effecting such change.)
>
> Good luck.
>
> >I hope that the above answers the
> >bunch of questions you posed above. Please let me know if some points
> >are not yet adequately covered.
>
> They do. I understand your position. I just have no further interest in
> debating it.

Then of course you could keep silence, which is an alternative I
pointed out in my previous post (quoted below).

> >> But as I said in Usenet, I really don't want to carry on this
> >> conversation.
> >> It
> >> is not fun. You are not listening, either to me or to others on the
> >> newsgroup.
> >> I feel like you are blaming me personally simply because you don't like
> >> what I am saying. If this kind of thing happens socially, I generally
> >> walt
> >> away from the offending person. I did so, and you have followed me and
> >> sent me personal mail. So I am talking again. If I still find the
> >> conversation
> >> unpleasant, I will walk away again.
> >
> >It is my humble opinion that people engage in scientific discussions
> >because they desire to find the truth and not becuase they desire
> >to find fun and pleasure.
>
> That is the difference. You are engaging in a scientific discussion. I am
> merely chatting on Usenet. We are invested in differing amounts.

May I remind you that for chatting there is IRC. Why does sci.crypt
have the prefix 'sci' and not 'talk'??? (Are you formally asking
here others not to take seriously of whatever you say in the group???)

> >In order to find the truth one is ready to
> >pay the price, if necessary, of unpleasantness. I am excluding
> >impoliteness here which I hate and which unfortunately I experienced
> >often in discussion groups. I can assure you that from my standpoint
> >I have at no time point blamed you (in the sense of the word 'blame'
> >as I understand it). But it is true that I am very hard-necked (is
> >this a correct English word?) in scientific discussions.
>
> "Stiff necked" is proper. But nice choice.
>
> >If something
> >is in my opinion wrong, I alway say it 'very' 'very' directly (without
> >'speaking through the flowers'). Perhaps you are not used to
> >discussion partners of my kind.
>
> I am.
>
> >After all, to take part in a scientific
> >discussion is free will, there is no obligation. If one gets tired,
> >feel the stuff uninteresting, or for whatever reason, one is
> >entirely free to stop arguing. You are not bound to reply to posts
> >of anybody. Due to my hard-neckedness I used to continue discussion,
> >however, up to the very end. (Not very long ago I was engaged in a
> >discussion on possible parapsychological influences on phsical events
> >and in a discussion on Rivest's chaffing and winnowing. Both cases had
> >cost me quite a lot of extra time and energy because a few discussion
> >partners were not arguing scientifically in my humble view.)
>
> That's fine. I respect this.
>
> >> >The second has the title 'US5835600 Block encryption algorithm with
> >> >data dependent rotations' and is a patent issued to Prof. R. Rivest.
> >> >This is interesting in that only three days after the issue of
> >> >the patent it is already to be found on a Web page maintained by some
> >> >private person, (highly probably) in contradiction to the thesis
> >> >that 'Almost all patents are examined by almost nobody'.
> >>
> >> I knew about it the day it was released. This was important. Again, I
> don't
> >> see how it relates to a discussion as whether or not patents are
> >> generally

> >> read by academics. They are not. My apologies if you don't like this
> >> fact.
> >
> >You misunderstood me. My point here is not concerned with the academics.
> >The point is that patents do get attention from the scientific
> >(professional) community (which is not identical to the set of
> >academics!).
>
> Of course they get attention from the scientific community; I didn't think
> that
> was an issue. If I misquoted you, it is because I could think of no other
> reason
> for you to bring the item up. If you simply brought it up to make the
> point
> that patents get attention from professionals, then I agree with you.

I don't care whether the academics choose to ignore something, so long
as the MAJORITY of the scientific community pay attention to that.

> I do not believe that "get attention from" and "are worth an
> academic's time to read" are very different things.

I refer you again to what I said about 'academics'.

> >> >I think that it is universally true that where big money and/or its
> >> >equivalents, e.g. personal survival, are involved there will be
> >> >proportionately high attention paid by the community. Otherwise
> >> >it could well happen under circumstances that very few people read
> >> >a very high quality scientific paper that has no practical relevances.
> >> >This is lamentable but is a fact of life.
> >>
> >> Yes, and the academic community is generally unconcerned with patents.
> >> The examples above are from the business community.

So Prof. Rivest is one of the business community, not scientist,
according to you???

> >You seem to stick to the issue of academic community. I am concerned
> >with the scientific (professional) community, including the
> >scientists that work for the business firms. Much of the divergence
> >of our discussions can be traced to this fact. Since most of
> >the readers of the group are not academics, not to say having tenures,
> >I believe that my broader standpoint is the more appropriate one
> >for the present discussion than yours.
>
> Oh. I didn't think this was about the professional community.
> I thought we
> were talking about patents being 1) worthy of academic citation, 2) worthy
> of "publication" status on par with real publications, and 3) the recipient
> of peer review similar to real publications.

There is really only one single criterion for any publication: Worthy
to be known by the community it addresses! A paper is NOT printed
only for reading by those having tenures!!! And the scientific
community isn't comprised of teenagers who are incapable of knowing
what are good stuffs and what are bad and consequently have to follow
strictly the advice and guides (or commands) of professors as to what
they should and should not read!!!

> >> >BTW, although I haven't yet closely studied Rivest's patent, I guess
> >> >that there is a certain (maybe only weak) parallel of a small part
> >> >of his idea with a small part of the idea underlying my WEAK3-E, since
> >> >I also make use of rotations in block encryption (that is controled
> >> >by a hash value which is dependent on the plain text being processed
> >> >and hence data dependent (though differing in the detailed manner
> >> >from the patent)).
> >>
> >> And there's an algorithm in my book that has data dependent rotations.
> >> And IBM makes a claim about them.
> >>
> >> Again, I don't see what this has to do with the fact that patents are
> >> not
> >> generally cited as academic papers, are not generally considered
> publications
> >> by academics, and are not subjected to the same peer review process
> >> as academic papers.
> >
> >You misunderstood me. Actually here it is my fault. I wanted to put
> >before the string 'BTW' above the phrase 'Something off-topic:'.
> >But an uncontrolled movement of my hand caused the message to be
> >sent without that phrase. My intention here is more 'personal'
> >(or egoistic), hoping that someone of the group would say something
> >about the idea of using rotations (which I also used in my algorithm
> >and of which I like to hear some opinions for eventually improving
> >my own design).
>
> Oh.

Isn't that even without the forgotten phrase it is fairly evident
from the wordings that I was not arguing for patent or not patent
in this particular paragraph but speaking on a possible comparison

of the basic ideas underlying the two algorithms?

> >I hoped that I have given you a sufficiently understandable response.
> >If you have further questions or points, I shall be very glad to
> >answer them and discuss with you. As I said, I am very hard-necked in
> >scientific discussions (not so in private social discussions) and
> >I express my thougts plainly without 'artificial' modifications.
> >I hope I have not said anything impolite, using bad words etc.,
> >since I have not a single reason to be angry with you. Maybe
> >some words were 'strong' because I don't like to 'speak through
> >the flowers' but I am not conscious of having ever said anything that
> >does not correspond to plain facts. Please feel free to point out
> >directly, quoting my writing, if this is not true in your opinion,
> >so that I may learn something in this respect.

Cheers,

M. K. Shen

Subject: Re: Memo to the Amateur Cipher Designer
Date: Thu, 19 Nov 1998 08:57:45 +0100
From: Mok-Kong Shen <mok-kong.shen@stud.uni-muenchen.de>
Message-ID: <3653CF79.B04B3E6B@stud.uni-muenchen.de>
References: <19981118100730.12389.00002976@ng104.aol.com>
<3652C9E3.1B972124@stud.uni-muenchen.de>
Newsgroups: sci.crypt
Lines: 17

JPeschel wrote:
>

>
> And the discussions should be kept outside of the group.
> The truth is, Mok, if you want to read patents go ahead
> and read them. Nobody is going to stop you.
>
> But please stop yammering about them here.

I way not even saying that I myself am reading much patents or
not. I was only arguing about the VALUE of patent documents
which Bruce Schneier negated, saying that these are not even
publications. I am not yammering, nor chatting, which Bruce
Schneier said he is doing.

M. K. Shen

Subject: Re: Memo to the Amateur Cipher Designer
Date: Thu, 12 Nov 1998 14:02:26 GMT
From: "Joseph K. Nilaad" <jknilaad@xoommail.com>
Message-ID: <364AEA72.559E@xoommail.com>
References: <3649DD17.867829C@stud.uni-muenchen.de>
Newsgroups: sci.crypt
Lines: 45

Mok-Kong Shen wrote:
>
[snip]
> Let me say some more words against the be-littlement of patent
> publications. I'll choose an analogy which I already have used.
> DES is described in your well-known book. But where did you get
> the informations? Maybe you got that from another author. But
> then where did he get that? Ultimately one comes to the original
> government publication, if one continues asking. So if that original
> document (which is perhaps not a scientific paper in the eyes of
> certain academics) doesn't get cited very often, that doesn't mean
> anything, in particular it does not affect the scientific value and
> significance (contribution) of that document. If a patented crypto
> is really good, it will be popular and more people will study it,
> eventually publishing papers on it. Whether the authors of the
> papers cite the patent document is in my view not very essential.
> One thing is on the other hand certain, namely that without the
> publishing of the original document these papers could not exist.
You have very good point. Each patent here in the U.S. usually has
related patents that are searched against. Though, I think it's kind of
narrow but again 1000's of patent applications are filed yearly. I've
heard that 2-3 years back-log.

Back to crypto. I agreed with your statement "If a patented crypto
is really good, it will be popular and more people will study it". I
think that Bruce will also agree in which he has posted a reply to me
here in sci.crypt. But he does have a valid point in the sense that why
should one work for someone for free? The patents are owned by
somebody! In case of DES, no one yet is being sued by DES owner but
that doesn't guarantee the users of DES will not be sued. I think the
patent owners
should have choice to decide who can use their patents. I've seen some
shareware allow for personal use. Maybe patent owners will do the same.

We all agree that we want the *BEST CRYPTO* possible, patent or not,
academic or not, etc. Ultimately, we need our own privacy.

We can drag this forever. I sometime think, are we the muppets being manipulated?

Joseph K. Nilaad
Nature is simple and beautiful...
Life is too short to appreciate it all...

Subject: Re: Memo to the Amateur Cipher Designer
Date: Mon, 16 Nov 1998 03:07:07 GMT
From: Denning Langston <denninglangston@yahoo.com>
Message-ID: <364F961A.2408B9D0@yahoo.com>
References: <3649DD17.867829C@stud.uni-muenchen.de>
Newsgroups: sci.crypt
Lines: 45

Mok-Kong Shen wrote:

> As I wrote previously I know that many organic
> chemical compounds, for example, are patented. A competitor can't
> use these or have to pay license fees. That's why large chemical
> firms need people knowledgeable in such patents in order that
> they can do their business properly. These companies have so much
> money that the court expenses are really entirely negligible. This
> is one example that I happen to be able to present. Are you going
> to counter with sentence like 'But in crypto it is different'?
>
> M. K. Shen

This is entirely wrong, at least in the US.

Organic compounds are not patentable in and of themselves. Processes that create useful chemical compounds efficiently or cheaply are patentable, and specific uses of chemical compounds are patentable (pharmaceuticals, pesticides, herbicides, etc.), but chemical compounds in and of themselves are not patentable.

Perhaps you are confusing chemical compounds whose synthesis is patented and that synthesis is the only known method by which the compound can be created. This will render the compound proprietary because no-one can find a unpatented process that results in the same compound. When the patent runs out, the entire world starts making it. (polytetraflouroethylene is such an example - aka Teflon when DuPont had the patent, PTFE now that they don't.)

The beginning of my career as a process design chemical engineer soley consisted of breaking both foreign and domestic process patents. Analyze a process resulting in compound 'A', determine it's patentable 'uniqueness', and devise a process that results in the same compound without utilizing the same uniqueness. This usually resulted in a new patentable chemical process, at which point 'they' (the competition) would do the same to us. It's a chemical game of cat 'n mouse.

Occasionally a court case would develop when a patent was granted for a chemical synthesis process that was not unique, but that rarely occurred (and at least one person lost his job when it did!).

Denning Langston, PE

Subject: Re: Memo to the Amateur Cipher Designer
Date: Mon, 16 Nov 1998 08:51:35 +0100
From: Mok-Kong Shen <mok-kong.shen@stud.uni-muenchen.de>
Message-ID: <364FD987.C11A2589@stud.uni-muenchen.de>
References: <364F961A.2408B9D0@yahoo.com>
Newsgroups: sci.crypt
Lines: 42

Denning Langston wrote:

> This is entirely wrong, at least in the US.
>
> Organic compounds are not patentable in and of themselves. Processes that
> create useful chemical compounds efficiently or cheaply are patentable, and
> specific uses of chemical compounds are patentable (pharmaceuticals,
> pesticides, herbicides, etc.), but chemical compounds in and of themselves
> are not patentable.
>
> Perhaps you are confusing chemical compounds whose synthesis is patented and
> that synthesis is the only known method by which the compound can be
> created. This will render the compound proprietary because no-one can find a
> unpatented process that results in the same compound. When the patent runs
> out, the entire world starts making it. (polytetraflouroethylene is such an
> example - aka Teflon when DuPont had the patent, PTFE now that they don't.)

I am not a chemist. You certainly exactly about chemical patents.
But your argument does not invalidate the essence of my arguments
in this thread, which is that patents (more properly the informations
contained and made public through the patents) contain (at least
in part) essential and valuable scientific informations which

should not be ignored by the academics (those at the universities and the academies of sciences) and, as far as I can make out, are indeed not largely ignored by them (the converse was argued by Bruce Schneier.) What I can find about chemistry is that there are lots of literature references to patents, see e.g.

    Ullmann's Encyclopedia of Industrial Chemistry, 5th Ed.
    VCH Verlagsgesellschaft, Weinheim, 1985.

Your information on the issue of whether a compound or only the synthesis method is patentable reminds me of a long disputed and more deep issuue whether only a machine or also simply an idea without being coupled to a machine design is patentable. More specifically in the interest of this thread, the question is whether an algorithm is patentable or is it necessary to present for patent application some hardware implementing it. I am not knowledgeable about this. But examples like the RSA patents seem to indicate that current law practices do allow ideas to be patentable to some extent.

M. K. Shen

Subject: Re: Memo to the Amateur Cipher Designer
Date: Tue, 10 Nov 1998 15:34:48 GMT
From: "Joseph K. Nilaad" <jknilaad@xoommail.com>
Message-ID: <36485D18.4093@xoommail.com>
References: <363F44ED.FEF5E3C@stud.uni-muenchen.de>
Newsgroups: sci.crypt
Lines: 118

Bruce Schneier wrote:
>
> On Tue, 03 Nov 1998 19:01:17 +0100, Mok-Kong Shen
> <mok-kong.shen@stud.uni-muenchen.de> wrote:
[snip]
>
> I believe you would be amazed by what gets through the patent office.
> The only thing they regularly catch are perpetual motion machines; bad
> cryptography will fly right over their heads. (For heavens sake, they
> can't even week out impossible compression patents.)
I just can't help it to get in this thread, first of all, I don't work at patent office. It is unfair that the way you judge their works. 1000's patents have been filed yearly, how many people are working there. Give them a break. In crypto, you know darn well it takes a lot of time to analyze, for instance, whatever FISH you have, have you finished with it? Maybe those people at the patent office are not expert in any field. But just because they are not *expert* in crypto, it doesn't mean that they are not in the other field. Now why don't you tell me the merits of turbine engine and rotary engine?

>
> >I haven't submitted patents. But it is certainly permitted that I
> >discuss about the matter? In the real world there are also judges
> >who are incapable and decide wrongly. As I argued previously patents
> >are NOT issued simply because the examiners employed by the patent
> >officices think they are o.k.
>
> Yes. You argued that previously. You are wrong. Patents are issued
> because the patent examiner who has the application has allowed some
> of the claims.
Perhaps you should do a thorough research on how patent is granted before you make this kind of statement. Are there any patent guys out there want to defend yourselves? As far as I know there are more than one person who make decision whether to grant or not.

[snip]
>
> >(Recently in a mailing list someone
> >said he could not exploit the idea of using faces in a scheme for
> >passphrase entry because IBM has a patent that is very broad to
> >cover that.) In which sense is such a public review less effective
> >(stringent) than a 'peer review' in a journal?
>
> It's not public. There is no such public review. In very competitive
> industries--pharmasuticals come to mind--companies watch foreign
> filings for clies as to what the competition is doing. But I know of
> no instance of a company trying to block a patent from being awarded.
Agree.

[snip]
> >> THen I won't be doing my job, because this discussion isn't worth
> >> that much time. Sorry; I don't mean to be rude. I read sci.crypt for
> >> fun, not to find more work to do.
I tend to incline that your comments are begining to amuse me.

> >
> >I don't see you are responding to my point here at all. I claimed that
> >the quoted challenge is inappropriate. What has that to do with
> >fun or not fun in sci.crypt or work or not work?? You challenged other
> >people to consult lawers. That is not only WORK but monetary expenses!!
>
> Look, I didn't challenge anyone to consult a lawyer. I suggested that
> if you don't believe me, you should consider asking someone else who
> may know. I really don't care enough about this argument to spend the

> time necessary to convince you.
>
> And there are some excellent books on patent law by Nolo Press.
Different countries have different laws, so what good is it for Nolo
Press in other countries. It is unfair challenge.

[snip]

My point is that it doesn't matter whether it is amateur or expert who
design the crypto, patent or not, we all want the best crypto possible.
If AES confine to non patent algorithm, I think it is very narrow
minded. So what if the best algorithm we have to pay to use it, so be
it. If the guy who own the best algorithm, has patent on it and charge
too much for use, then most likely there will be less people using his
crypto.

AES closed the for entries this year, the winner will be annouced few
years later. Get real. I don't know exactly how many entries, but I
know it is less than 20. Let say 2 years to announce the winner, and
3-5 persons "so called crypto expert" assigned to each algorithm to do
cryptanalysis. How much money are we talking about? Not many people
can afford to work for charity for that long. Maybe you can.

Oh, you guys offer $10,000 for a person who has the best cryptanalysis
against TWOFISH, that is a smart way to spend money. Let see if 5
people try for 5 months that 25 man-months, divide into $10,000 that
$400 a month per person. The more people try, the less money per
month. If everybody is *considered* failure to come up the answer, zero
out of pocket. *FREE LABOR*. Wow, that is novelty. $10,000 for the
expert, they don't even blink at it. Perhaps you can get Casio to
sponsor your algorithm up the ante, then maybe Mr. Ritter or Dave Scott
will give a shot at it.

To me, whether the crypto algorithm is published or not, it is
irralevant. If the algorithm has any merits, let it stands out! I've
worked with many people who pubished so many papers, but it seems that's
all they can do. When it comes to do real work, most of the time, they
can't even walk and chewing gum at the same time. Just because you (in
general) can write, it doesn't mean your stuff is correct or better than
anyone else.

In addition, it is lucky for you that you can use DDJ for you contest,
but not for many like Mr. Ritter or David A. Scott, etc.

This thread is tooooo long, Bruce, what do you have against patent
algorithms, sum it all out, don't waste any bandwidth.

I am beginning to find this thread very amusing now, ha ha ha.

Joseph K. Nilaad
Nature is simple and beautiful...
Life is too short to appreciate it all...

Subject: Re: Memo to the Amateur Cipher Designer
Date: 10 Nov 1998 11:53:37 -0500
From: juola@mathcs.duq.edu (Patrick Juola)
Message-ID: <729r2h$91c$1@quine.mathcs.duq.edu>
References: <36485D18.4093@xoommail.com>
Newsgroups: sci.crypt
Lines: 41

In article <36485D18.4093@xoommail.com>,
Joseph K. Nilaad <jknilaad@xoommail.com> wrote:
>Bruce Schneier wrote:
>>
>> On Tue, 03 Nov 1998 19:01:17 +0100, Mok-Kong Shen
>> <mok-kong.shen@stud.uni-muenchen.de> wrote:
>[snip]
>>
>> I believe you would be amazed by what gets through the patent office.
>> The only thing they regularly catch are perpetual motion machines; bad
>> cryptography will fly right over their heads. (For heavens sake, they
>> can't even week out impossible compression patents.)
>I just can't help it to get in this thread, first of all, I don't work
>at patent office. It is unfair that the way you judge their works.
>1000's patents have been filed yearly, how many people are working
>there. Give them a break. In crypto, you know darn well it takes a lot
>of time to analyze, for instance, whatever FISH you have, have you
>finished with it? Maybe those people at the patent office are not
>expert in any field. But just because they are not *expert* in crypto,
>it doesn't mean that they are not in the other field. Now why don't you
>tell me the merits of turbine engine and rotary engine?

But this whole thread started with a discussion of whether or not
patents were "peer-reviewed." They're not, as you admit above.
The fact that patent agents aren't experts in cryptography isn't a
moral failing -- or even surprising -- as you point out, they
can't be expert in *everything*.

But this doesn't mean that they're the "peers" of the experts who
constitute the peer reviewers. Quite the contrary, it's an
assertion that they're not -- and as such, the opinion of a patent

agent doesn't constitute "peer review." Furthermore, the opinion of a patent agent is given on other grounds than scientific merit and for a different purpose.

No apologies for patent agents should be necessary -- they're overworked civil servants doing the best they can under adverse conditions. But they're certainly not the peers of the authors of papers at CRYPTO'97.

-kitten

Subject: Re: Memo to the Amateur Cipher Designer
Date: Tue, 10 Nov 1998 17:58:27 +0100
From: <tbb03ar@mail.lrz-muenchen.de>
Message-ID: <Pine.GSO.4.03.9811101744230.21447-100000@sun5.lrz-muenchen.de>
References: <36485D18.4093@xoommail.com>
Newsgroups: sci.crypt
Lines: 41

On Tue, 10 Nov 1998, Joseph K. Nilaad wrote:

> Bruce Schneier wrote:
> >
> > On Tue, 03 Nov 1998 19:01:17 +0100, Mok-Kong Shen
> > <mok-kong.shen@stud.uni-muenchen.de> wrote:
> [snip]
> >
> > I believe you would be amazed by what gets through the patent office.
> > The only thing they regularly catch are perpetual motion machines; bad
> > cryptography will fly right over their heads. (For heavens sake, they
> > can't even week out impossible compression patents.)
> I just can't help it to get in this thread, first of all, I don't work
> at patent office. It is unfair that the way you judge their works.
> 1000's patents have been filed yearly, how many people are working
> there. Give them a break. In crypto, you know darn well it takes a lot
> of time to analyze, for instance, whatever FISH you have, have you
> finished with it? Maybe those people at the patent office are not
> expert in any field. But just because they are not *expert* in crypto,
> it doesn't mean that they are not in the other field. Now why don't you
> tell me the merits of turbine engine and rotary engine?
> ...

Nobody says the people at the patent office wouldn't do a good job.

Of course they do and of course what they do is importgant for all of us:
Without patents all new inventions would have to be kept secret to keep
others from copying it.

But it is really not their job to test an encryption algorithm for
strength.

You should have a look at what strange mashines - and algorithms - were
patented. It's interesting and funny.

Andreas Enterrottacher

enterrottacher@lrz.tu-muenchen.de
enterrottacher@t-online.de

Subject: Re: Memo to the Amateur Cipher Designer
Date: 10 Nov 1998 18:16:26 GMT
From: aph@cygnus.remove.co.uk (Andrew Haley)
Message-ID: <729vtq$5oq$1@korai.cygnus.co.uk>
References: <36485D18.4093@xoommail.com>
Newsgroups: sci.crypt
Lines: 25

Joseph K. Nilaad (jknilaad@xoommail.com) wrote:

: My point is that it doesn't matter whether it is amateur or expert who
: design the crypto, patent or not, we all want the best crypto possible.
: If AES confine to non patent algorithm, I think it is very narrow
: minded. So what if the best algorithm we have to pay to use it, so be
: it.

If the AES is to be universally used, it must not be encumbered by
royalties. Much of the software which runs the Internet is free, and
so there is no possibility of the supplier paying a royalty.

Any algorithm which can only be used in unfree software faces an
enormous barrier to acceptance: such an algorithm had better be
obviously better than any free algorithm or it will not be used.

: If the guy who own the best algorithm, has patent on it and charge
: too much for use, then most likely there will be less people using
: his crypto.

Indeed. And the AES will have failed, and people will carry on using
a diverse bunch of algorithms. The whole selection procedure will
have been a complete waste of time.

Andrew.

Subject: Re: Memo to the Amateur Cipher Designer
Date: Wed, 11 Nov 1998 16:35:20 -0600
From: jgfunj@EnqvbSerrGrknf.pbz (W T Shaw)
Message-ID: <jgfunj-1111981635210001@dialup164.itexas.net>
References: <729vtq$5oq$1@korai.cygnus.co.uk>
Newsgroups: sci.crypt
Lines: 16

In article <729vtq$5oq$1@korai.cygnus.co.uk>, aph@cygnus.remove.co.uk
(Andrew Haley) wrote:
>
> Indeed. And the AES will have failed, and people will carry on using
> a diverse bunch of algorithms. The whole selection procedure will
> have been a complete waste of time.
>
AES has several implications, only one of them be that could replace lots
of others. It is destined not to do that, so consider that having a
government standard is still necessary, at least for them. The AES
process has added to the mix of ciphers, and more is better.
--

The public is harder to steamroller than some might think.

Decrypt with ROT13 to get correct email address.

Subject: Re: Memo to the Amateur Cipher Designer
Date: Tue, 10 Nov 1998 18:25:43 GMT
From: schneier@counterpane.com (Bruce Schneier)
Message-ID: <364982dc.1295475@news.visi.com>
References: <36485D18.4093@xoommail.com>
Newsgroups: sci.crypt
Lines: 127

On Tue, 10 Nov 1998 15:34:48 GMT, "Joseph K. Nilaad"
<jknilaad@xoommail.com> wrote:

>Bruce Schneier wrote:
>>
>> On Tue, 03 Nov 1998 19:01:17 +0100, Mok-Kong Shen
>> <mok-kong.shen@stud.uni-muenchen.de> wrote:
>[snip]
>>
>> I believe you would be amazed by what gets through the patent office.
>> The only thing they regularly catch are perpetual motion machines; bad
>> cryptography will fly right over their heads. (For heavens sake, they
>> can't even week out impossible compression patents.)
>
>I just can't help it to get in this thread, first of all, I don't work
>at patent office. It is unfair that the way you judge their works.
>1000's patents have been filed yearly, how many people are working
>there. Give them a break. In crypto, you know darn well it takes a lot
>of time to analyze, for instance, whatever FISH you have, have you
>finished with it? Maybe those people at the patent office are not
>expert in any field. But just because they are not *expert* in crypto,
>it doesn't mean that they are not in the other field. Now why don't you
>tell me the merits of turbine engine and rotary engine?

I'm sorry. I thought I was giving them a break. I know it is
difficult. I know that they do the best job they can. I know that
lots of things get by them. I don't mean to malign the patent office
at all. I apologise if you thought otherwise.

>Perhaps you should do a thorough research on how patent is granted
>before you make this kind of statement. Are there any patent guys out
>there want to defend yourselves? As far as I know there are more than
>one person who make decision whether to grant or not.

Generally, there is just one examiner per patent. Occasionally they
are bumbed to supervisors for review, but not often.

I understand that people aren't going to take my word here, but I
don't have the time and patience to provide evidence to convince.
Anyone who is interested is welcome to do their own research.

>Different countries have different laws, so what good is it for Nolo
>Press in other countries. It is unfair challenge.

Don't know. I believe they discuss foreign filings And I don't mean
it as a challenge. I just suggested a book with information. People
are welcome to either read it or not, or to find their own books.

>My point is that it doesn't matter whether it is amateur or expert who
>design the crypto, patent or not, we all want the best crypto possible.
>If AES confine to non patent algorithm, I think it is very narrow
>minded. So what if the best algorithm we have to pay to use it, so be
>it. If the guy who own the best algorithm, has patent on it and charge
>too much for use, then most likely there will be less people using his
>crypto.

Possibly. NIST made their decision based on their own analysis and outside input. They decided to require submissions to be unpatented. Perhaps it is a mistake; we will find out soon enough.

>AES closed the for entries this year, the winner will be annouced few
>years later. Get real. I don't know exactly how many entries, but I
>know it is less than 20. Let say 2 years to announce the winner, and
>3-5 persons "so called crypto expert" assigned to each algorithm to do
>cryptanalysis. How much money are we talking about? Not many people
>can afford to work for charity for that long. Maybe you can.

We can't. One of the main problems with the whole process is that NIST is counting on everyone in the community to work on analysis for free. This has nothing to do with whether or not the algorithms are patented. I have no idea what level of cryptanalysis we will see by the Second AES Workshop. I hope we'll see some good work. My fear is that people are just too busy with real work.

NIST is hoping that because AES will be a world-wide standard that cryptanalysts will feel that it is in their best interest to donate their labor to the process.

>Oh, you guys offer $10,000 for a person who has the best cryptanalysis
>against TWOFISH, that is a smart way to spend money. Let see if 5
>people try for 5 months that 25 man-months, divide into $10,000 that
>$400 a month per person. The more people try, the less money per
>month. If everybody is *considered* failure to come up the answer, zero
>out of pocket. *FREE LABOR*. Wow, that is novelty. $10,000 for the
>expert, they don't even blink at it. Perhaps you can get Casio to
>sponsor your algorithm up the ante, then maybe Mr. Ritter or Dave Scott
>will give a shot at it.

Some submissions have large corporate sponsors. IBM, RSA, and NTT submitted algorithms. Intel funded the work on Serpent. But yes, you are right: NIST is asking for free labor from cryptanalysts.

>To me, whether the crypto algorithm is published or not, it is
>irralevant. If the algorithm has any merits, let it stands out! I've
>worked with many people who pubished so many papers, but it seems that's
>all they can do. When it comes to do real work, most of the time, they
>can't even walk and chewing gum at the same time. Just because you (in
>general) can write, it doesn't mean your stuff is correct or better than
>anyone else.

Agreed.

>In addition, it is lucky for you that you can use DDJ for you contest,
>but not for many like Mr. Ritter or David A. Scott, etc.

Dr Dobbs publishes different algorithms from different people. The issue with Twofish also includes Panama, which is not an AES submission and is by other people entirely. And Dr Dobbs is not involved with the Twofish contest, although they did sponsor the Blowfish contest.

>This thread is tooooo long, Bruce, what do you have against patent
>algorithms, sum it all out, don't waste any bandwidth.

I have nothing against patented algorithms. People are welcome to patent algorithms. I see no reason to implement patented algorithms when there are unpatented alternatifves. This is just good economics. I see no reason to perform free analysis on patented algorithms unless there is a good reason to do so. This is simply the same "work for free for someone else's benefit" argument that you gave above. Other than that, patented algorithms are fine.

Bruce
****
Bruce Schneier, President, Counterpane Systems Phone: 612-823-1098
101 E Minnehaha Parkway, Minneapolis, MN 55419 Fax: 612-823-1590
Free crypto newsletter. See: http://www.counterpane.com

Subject: Re: Memo to the Amateur Cipher Designer
Date: Wed, 11 Nov 1998 14:32:50 GMT
From: malinov@mindless.com
Message-ID: <72c76i$m3o$1@nnrp1.dejanews.com>
References: <364982dc.1295475@news.visi.com>
Newsgroups: sci.crypt
Lines: 26

Bruce said something like . . .

> Generally, there is just one examiner per patent. Occasionally they
> are bumbed to supervisors for review, but not often.

Last I checked, cryptography is examined at the USPTO in art unit 2766 by three primary examiners, two juniors on the verge of becoming primaries and four juniors still in their first six months. The senior five examine their cases independently. The new juniors report their cases to their supervisor. All AU2766 examines is crypto, although that includes every system (tv, phone, computer, network, ATM, etc.) which uses crypto in some way. There has been much talk about reorganizing the art.

With less than eighteen hours to examine each case, the patent office is only a filter. Bad patents are bound to get through although many more are caught and squashed. A patent can't rise to the level of "seal of approval," much less a peer review. It's just a property deed. Some of the paper is bound to represent worthless swamp land. It's up to the market to assign value.

David Cain

--
Power belongs to those who dare . . . Sapere Aude

----------== Posted via Deja News, The Discussion Network ==----------
http://www.dejanews.com/ Search, Read, Discuss, or Start Your Own

---

Subject: Re: Memo to the Amateur Cipher Designer
Date: Thu, 12 Nov 1998 22:00:05 GMT
From: ritter@io.com (Terry Ritter)
Message-ID: <364b5a51.14580571@news.io.com>
References: <364982dc.1295475@news.visi.com>
Newsgroups: sci.crypt
Lines: 60

On Tue, 10 Nov 1998 18:25:43 GMT, in <364982dc.1295475@news.visi.com>,
in sci.crypt schneier@counterpane.com (Bruce Schneier) wrote:

>[...]
>I see no reason to implement patented algorithms
>when there are unpatented alternatifves. This is just good economics.

This is *temporary* economics. By failing to compensate the work actually performed, we fail to build a profit-based business of cipher design. We have ciphers, yes. But we do not have a continuing business of cipher design, along with the expensive expertise and corporate history associated with other technologies.

>I see no reason to perform free analysis on patented algorithms unless
>there is a good reason to do so.

And that is a fundamental difference: Some people make what they know their property (even though most of what we all know is gained from open sources), and they protect their knowledge-property with trade secrecy. In contrast, patents are about *exposing* knowledge.

So on the one hand we have patents which *reveal* what they know, and on the other we have trade secrecy which *hides* what it knows from society -- unless they pay a fee, of course. Or unless they buy a book (for which they pay a fee).

It is true that a patent is a limited-term monopoly. But that monopoly costs users only to the extent that it successfully competes -- royalties and all -- in the marketplace against other solutions. The people can thus decide what to use, based on full prior knowledge of the costs and advantages.

Currently cryptanalysts *do* get fees to expose their private information, but cipher designers do *not* get rewards for making their information open to society. This would seem to be an interesting take on "freedom of information."

>This is simply the same "work for
>free for someone else's benefit" argument that you gave above. Other
>than that, patented algorithms are fine.

The idea of AES is to convince cipher designers to give away their work for free to companies who will turn around and sell it for a profit. Oh, yes, a few users will use free cipher implementations and so avoid payment. But most will not, so, in general, society will pay companies for merely *implementing* what they did not have to fund in research or development. AES is not about free end-user crypto; AES is about free crypto designs for companies to use to their profit.

In this way our government avoids compensating cipher design, since a continuing business of cipher design is seen as a threat.

Terry Ritter ritter@io.com http://www.io.com/~ritter/
Crypto Glossary http://www.io.com/~ritter/GLOSSARY.HTM

Subject: Re: Memo to the Amateur Cipher Designer
Date: Fri, 13 Nov 1998 09:02:29 +0100
From: <tbb03ar@mail.lrz-muenchen.de>
Message-ID: <Pine.GSO.4.03.9811130832040.6919-100000@sun5.lrz-muenchen.de>
References: <364b5a51.14580571@news.io.com>
Newsgroups: sci.crypt
Lines: 73

On Thu, 12 Nov 1998, Terry Ritter wrote:

>
> On Tue, 10 Nov 1998 18:25:43 GMT, in <364982dc.1295475@news.visi.com>,
> in sci.crypt schneier@counterpane.com (Bruce Schneier) wrote:
>
> >[...]
> >I see no reason to implement patented algorithms
> >when there are unpatented alternatifves. This is just good economics.
>
> This is *temporary* economics. By failing to compensate the work
> actually performed, we fail to build a profit-based business of cipher
> design. We have ciphers, yes. But we do not have a continuing
> business of cipher design, along with the expensive expertise and
> corporate history associated with other technologies.
>

Free software has a long tradition and no other software is developed
faster and more continuous than free software.

I don't see why ciphers shopuld be different: They are a neccessary part
of information security, but there are enough free ones.

>
> >I see no reason to perform free analysis on patented algorithms unless
> >there is a good reason to do so.
>
> And that is a fundamental difference: Some people make what they know
> their property (even though most of what we all know is gained from
> open sources), and they protect their knowledge-property with trade
> secrecy. In contrast, patents are about *exposing* knowledge.

I think the question was something completely different: Free algorithms
will be used by more people and they will be tested as well by some of the
implementors as by some of the users. Patented algorithms will be used by
few people - maybe only in the security systems of the developer. In the
worst case nobody else will test the cipher and the strength will be
unknown.

> ...
>
> Currently cryptanalysts *do* get fees to expose their private
> information, but cipher designers do *not* get rewards for making
> their information open to society. This would seem to be an
> interesting take on "freedom of information."
>

A good designer is a good cryptanalyst :-)

>
> >This is simply the same "work for
> >free for someone else's benefit" argument that you gave above. Other
> >than that, patented algorithms are fine.
>
> The idea of AES is to convince cipher designers to give away their
> work for free to companies who will turn around and sell it for a
> profit.

So you don't think the person or company developing the final AES will
earn lots of money because they are the developers of AES?

It is simple to make use of the patent rights if it is refused.
This way the developer gets either a good analysis of his patented
algorithm or he becomes the developer of AES.

No risk, as far as I can see.

Andreas Enterrottacher

enterrottacher@lrz.tu-muenchen.de
enterrottacher@t-online.de

Subject: Re: Memo to the Amateur Cipher Designer
Date: 13 Nov 1998 13:06:53 GMT
From: aph@cygnus.remove.co.uk (Andrew Haley)
Message-ID: <72hatd$9g3$1@korai.cygnus.co.uk>
References: <364b5a51.14580571@news.io.com>
Newsgroups: sci.crypt
Lines: 53

Terry Ritter (ritter@io.com) wrote:

: On Tue, 10 Nov 1998 18:25:43 GMT, in <364982dc.1295475@news.visi.com>,
: in sci.crypt schneier@counterpane.com (Bruce Schneier) wrote:

: >[...]
: >I see no reason to implement patented algorithms
: >when there are unpatented alternatifves. This is just good economics.

: This is *temporary* economics. By failing to compensate the work
: actually performed, we fail to build a profit-based business of cipher
: design. We have ciphers, yes. But we do not have a continuing

: business of cipher design, along with the expensive expertise and
: corporate history associated with other technologies.

: The idea of AES is to convince cipher designers to give away their
: work for free to companies who will turn around and sell it for a
: profit. Oh, yes, a few users will use free cipher implementations and
: so avoid payment.

You don't seem to be addressing what I think is the central point. A
successful AES candidate must be universal. This means that it must
be used everywhere, in both free and unfree software. A patented
algorithm may not be used in free software, so cannot be used
universally. Therefore a patented AES will fail to be universal.

I can see no point in having a standard cipher which is not universal.

: But most will not, so, in general, society will pay companies for
: merely *implementing* what they did not have to fund in research or
: development. AES is not about free end-user crypto; AES is about
: free crypto designs for companies to use to their profit.

AES is about a universal crypto standard, just like DES.

: In this way our government avoids compensating cipher design, since a
: continuing business of cipher design is seen as a threat.

There is no need for your government to compensate cipher designers.

The idea of the AES, as I see it, is to exchange the prospect of
future royalties for the advantage of having your algorithm approved.
This is what IBM accepted when the DES was standardized.

Andrew.

Subject: Re: Memo to the Amateur Cipher Designer
Date: 13 Nov 1998 23:26:59 GMT
From: olson@umbc.edu (Bryan G. Olson; CMSC (G))
Message-ID: <72if83as1@news.umbc.edu>
References: <364b5a51.14580571@news.io.com>
Newsgroups: sci.crypt
Lines: 41

Terry Ritter (ritter@io.com) wrote:

: in sci.crypt schneier@counterpane.com (Bruce Schneier) wrote:

: >I see no reason to perform free analysis on patented algorithms unless
: >there is a good reason to do so.

: And that is a fundamental difference: Some people make what they know
: their property (even though most of what we all know is gained from
: open sources), and they protect their knowledge-property with trade
: secrecy. In contrast, patents are about *exposing* knowledge.
[...]
: Currently cryptanalysts *do* get fees to expose their private
: information, but cipher designers do *not* get rewards for making
: their information open to society. This would seem to be an
: interesting take on "freedom of information."

Here's how the cipher analysis game is played: If Bob works on
an algorithm for free, then if he finds a weakness he gets to
publish, and if he doesn't he never has reveal he tried. If
a company pays Bob to look at their cipher, the company will
invariably insist that Bob sign a non-disclosure, so he can't
reveal weaknesses he finds. They'll also want to say that
Bob failed to find any weakness if that's the case.

: >This is simply the same "work for
: >free for someone else's benefit" argument that you gave above. Other
: >than that, patented algorithms are fine.

: The idea of AES is to convince cipher designers to give away their
: work for free to companies who will turn around and sell it for a
: profit. Oh, yes, a few users will use free cipher implementations and
: so avoid payment.

Whey one buys an implementation, one pays for the implementation.
If you look at how patented ciphers are sold, you'll find that
companies charge for _both_ the implementation and the patent
rights. Buy BSAFE and you include it's DES code in your product
at no extra cost. RSA you still have to license.

--Bryan

Subject: Re: Memo to the Amateur Cipher Designer
Date: Fri, 13 Nov 1998 23:58:24 GMT
From: schneier@counterpane.com (Bruce Schneier)
Message-ID: <364dc730.2338361@news.visi.com>
References: <72if83as1@news.umbc.edu>
Newsgroups: sci.crypt
Lines: 22

On 13 Nov 1998 23:26:59 GMT, olson@umbc.edu (Bryan G. Olson; CMSC (G)) wrote:
>Here's how the cipher analysis game is played: If Bob works on
>an algorithm for free, then if he finds a weakness he gets to
>publish, and if he doesn't he never has reveal he tried. If
>a company pays Bob to look at their cipher, the company will
>invariably insist that Bob sign a non-disclosure, so he can't
>reveal weaknesses he finds. They'll also want to say that
>Bob failed to find any weakness if that's the case.

This has been my experience, working on both open and proprietary cryptography. Occasionally a company hires us to review an open cryptographic primitive, and allows us to publish our results. Hence, the work on SSL and Microsoft PPTP, and an analysis of IPSec that we are going to start as soon as the RFCs are published.

Bruce
****
Bruce Schneier, President, Counterpane Systems Phone: 612-823-1098
101 E Minnehaha Parkway, Minneapolis, MN 55419 Fax: 612-823-1590
Free crypto newsletter. See: http://www.counterpane.com

Subject: Re: Memo to the Amateur Cipher Designer
Date: 16 Nov 1998 03:22:57 GMT
From: jsavard@freenet.edmonton.ab.ca ()
Message-ID: <72o5qh$4qn$3@news.sas.ab.ca>
References: <364b5a51.14580571@news.io.com>
Newsgroups: sci.crypt
Lines: 30

Terry Ritter (ritter@io.com) wrote:
: This is *temporary* economics. By failing to compensate the work
: actually performed, we fail to build a profit-based business of cipher
: design.

Certainly, there are some egregious cases; that of Edward S. Hebern comes to mind.

But leaving that aside, you have a valid point. While I don't think one can expect the market to purchase something patented for a situation where, say, DES is fully satisfactory, it is reasonable to say that the absence of a thriving cryptography industry means that the ciphers available for use are not as strong, or as well-analyzed, as they might be.

It is a perennial problem that even those who do have a need for security persistently undervalue it.

Of course, looking at many of the designs that have originated in the academic sector, I also must confess that I am inclined to think that, as far as conventional symmetric-key cryptography is concerned, there is not that great a need for vastly more secure algorithms than those which currently exist, or which could be obtained by trivially scaling-up an existing algorithm.

Of course, impressive complexity is not a proof of security; but that goal, as I have noted, is one I do not believe to be realistic, as I suspect it is equivalent to solving the halting problem.

John Savard

Subject: Re: Memo to the Amateur Cipher Designer
Date: Wed, 11 Nov 1998 03:39:17 GMT
From: "Joseph K. Nilaad" <jknilaad@xoommail.com>
Message-ID: <364906E5.3520@xoommail.com>
References: <36485D18.4093@xoommail.com>
Newsgroups: sci.crypt
Lines: 79

Bruce Schneier wrote:
[snip]

> Generally, there is just one examiner per patent. Occasionally they
> are bumbed to supervisors for review, but not often.
>
> I understand that people aren't going to take my word here, but I
> don't have the time and patience to provide evidence to convince.
> Anyone who is interested is welcome to do their own research.
Agreed totally. I've seen you've taken a lot of heat, we all have short fuse every now and then.

>
> >Different countries have different laws, so what good is it for Nolo

> >Press in other countries. It is unfair challenge.
>
> Don't know. I believe they discuss foreign filings And I don't mean
> it as a challenge. I just suggested a book with information. People
> are welcome to either read it or not, or to find their own books.
Sorry, I missed your point.

[snip]
> Possibly. NIST made their decision based on their own analysis and
> outside input. They decided to require submissions to be unpatented.
> Perhaps it is a mistake; we will find out soon enough.
>
> >AES closed the for entries this year, the winner will be annouced few
> >years later. Get real. I don't know exactly how many entries, but I
> >know it is less than 20. Let say 2 years to announce the winner, and
> >3-5 persons "so called crypto expert" assigned to each algorithm to do
> >cryptanalysis. How much money are we talking about? Not many people
> >can afford to work for charity for that long. Maybe you can.
>
> We can't. One of the main problems with the whole process is that
> NIST is counting on everyone in the community to work on analysis for
> free. This has nothing to do with whether or not the algorithms are
> patented. I have no idea what level of cryptanalysis we will see by
> the Second AES Workshop. I hope we'll see some good work. My fear is
> that people are just too busy with real work.
>
> NIST is hoping that because AES will be a world-wide standard that
> cryptanalysts will feel that it is in their best interest to donate
> their labor to the process.
It seems obvious to me that the person who will be benefit the most
is the government. They are using more computers than any organization.
But looking at different view, this may save our tax dollars. For all
us average citizens, maybe AES doesn't matter much. What I want to see
is, our Bill of Rights are not violate. For community work, perhaps
it's time to use the famous phrase "ask not what can your country do
for you, but what you can do to your country" or something like that.

[snip]
>
> Some submissions have large corporate sponsors. IBM, RSA, and NTT
> submitted algorithms. Intel funded the work on Serpent. But yes, you
> are right: NIST is asking for free labor from cryptanalysts.
This really bothers me. I can see if all works are for a good sake.
Someone may be benefit from this, but all you guys get is piss in your
dark pants, you get the warm feelings but nobody notice (in the long
run though).

[snip]
>
> I have nothing against patented algorithms. People are welcome to
> patent algorithms. I see no reason to implement patented algorithms
> when there are unpatented alternatifves. This is just good economics.
> I see no reason to perform free analysis on patented algorithms unless
> there is a good reason to do so. This is simply the same "work for
> free for someone else's benefit" argument that you gave above. Other
> than that, patented algorithms are fine.
Now for all you guys out there, Bruce has made his point.

Can we now live in sci.crypt in harmony?

Joseph K. Nilaad
Nature is simple and beautiful...
Life is too short to appreciate it all...

Subject: Re: Memo to the Amateur Cipher Designer
Date: 30 Oct 1998 14:48:32 -0500
From: juola@mathcs.duq.edu (Patrick Juola)
Message-ID: <71d56g$p9e$1@quine.mathcs.duq.edu>
References: <363A0105.81B31DB5@stud.uni-muenchen.de>
Newsgroups: sci.crypt
Lines: 38

In article <363A0105.81B31DB5@stud.uni-muenchen.de>,
Mok-Kong Shen <mok-kong.shen@stud.uni-muenchen.de> wrote:
>Bruce Schneier wrote:
>>
>
>> Some cryptographic algorithms are patented, yes. I'm not sure how
>> that's relevent. I do not know of any academic cryptographers that
>> regularly look through the U.S. patent system. Patents are not a
>> peer-reviewed publication. If an academic (in any discipline)
>> presented a list of patents to his tenure review board, they would not
>> be considered publication.
>
>This is not true. Applications for patents are examined by a number
>of professionals in the corresponding fields to ensure that the
>ideas are really novel and useful.

They are not. They are examined by professional patent agents who
typically know something, but not very much, of the areas involved.

The existence of embarassingly large numbers of thoroughly ludicrous

patents is well-documented. The basic problem is that patent agents are *NOT* "experts" in the domains that they are passing judgement upon, and their opinions of what is "novel" are frequently wrong and completely misguided based on their unfamiliarity with the literature.

> There are huge data bases maintained
>by the patent offices and are carefully checked to ensure patents are
>not given to some one bring forth duplication or near duplication of
>prior art.

Based, of course, on other *patents*. But no such databases exist for publications.

And as the patent agents never (or rarely) read "the literature", they don't know about new developments.

-kitten

Subject: Re: Memo to the Amateur Cipher Designer
Date: Wed, 04 Nov 1998 11:59:43 -0600
From: jgfunj@EnqvbSerrGrknf.pbz (W T Shaw)
Message-ID: <jgfunj-0411981159440001@207.22.198.223>
References: <71d56g$p9e$1@quine.mathcs.duq.edu>
Newsgroups: sci.crypt
Lines: 26

In article <71d56g$p9e$1@quine.mathcs.duq.edu>, juola@mathcs.duq.edu (Patrick Juola) wrote:

>
> And as the patent agents never (or rarely) read "the literature",
> they don't know about new developments.
>
Cryptography is a special class, probably not entirely alone as other subjects might also get similiar attention. Consider what is apt to happen: PTO gets some crypto related application; it gets put into a pile since it might be messing with ideas that could classified. The application is properly forwarded to the Defense Department for review, if not directly to NSA itself.

It would be on the recommendations of the particular agencies that handled the details of inspection that the PTO office would act. The paperwork could be simply passed back quickly to PTO and processed as representing something trivial, but patentable all the same, returned and rejected to the submitter with no reasons given, or held for further study. From there, more options could click in. I bet you Ritter's work got lots more than a quick glance.
--

Remember...vote early and vote often ;)

Decrypt with ROT13 to get correct email address.

Subject: Re: Memo to the Amateur Cipher Designer
Date: Thu, 05 Nov 1998 00:00:35 GMT
From: dscott@networkusa.net
Message-ID: <71qpr3auf1@nnrp1.dejanews.com>
References: <jgfunj-0411981159440001@207.22.198.223>
Newsgroups: sci.crypt
Lines: 44

In article <jgfunj-0411981159440001@207.22.198.223>,
  jgfunj@EnqvbSerrGrknf.pbz (W T Shaw) wrote:
> In article <71d56g$p9e$1@quine.mathcs.duq.edu>, juola@mathcs.duq.edu
> (Patrick Juola) wrote:
>
> >
> > And as the patent agents never (or rarely) read "the literature",
> > they don't know about new developments.
> >
> Cryptography is a special class, probably not entirely alone as other
> subjects might also get similiar attention. Consider what is apt to
> happen: PTO gets some crypto related application; it gets put into a pile
> since it might be messing with ideas that could classified. The
> application is properly forwarded to the Defense Department for review, if
> not directly to NSA itself.
>
> It would be on the recommendations of the particular agencies that handled
> the details of inspection that the PTO office would act. The paperwork
> could be simply passed back quickly to PTO and processed as representing
> something trivial, but patentable all the same, returned and rejected to
> the submitter with no reasons given, or held for further study. From
> there, more options could click in. I bet you Ritter's work got lots more
> than a quick glance.
> --

  If the NSA is doing its job at all any encryption that is used at all on the net would be analyzed by them. I am sure megabucks where spent on PGP since it is so common. I think my stuff is stronger than IDEA or the FISHY methods but since not in public eye I doubt if they have given it as much attention yet. But it is very different

from the kind they are use to breaking. Since 19u is built around
19 bit boundaries on a PC indian type of machine. The program has
to be decrypt by several passes in the reverse direction. And as
the R of RSA is now pushing or talking about. It is a true all or
nothing encryption something that may be beyond the B S crypto
class of peoples current limited mind kind of thinking.

--
http://cryptography.org/cgi-bin/crypto.cgi/Misc/scott19u.zip
http://members.xoom.com/ecil/index.htm

----------== Posted via Deja News, The Discussion Network ==----------
http://www.dejanews.com/ Search, Read, Discuss, or Start Your Own

---

Subject: Re: Memo to the Amateur Cipher Designer
Date: Wed, 04 Nov 1998 16:41:25 -0800
From: aXcarol@apple.com (Andrew Carol)
Message-ID: <aXcarol-0411981641250001@andrew1.apple.com>
References: <71qpr3auf1@nnrp1.dejanews.com>
Newsgroups: sci.crypt
Lines: 36

In article <71qpr3auf1@nnrp1.dejanews.com>, dscott@networkusa.net wrote:

>I doubt
>if they have given it as much attention yet. But it is very different
>from the kind they are use to breaking. Since 19u is built around
>19 bit boundaries on a PC indian type of machine.

The effect the word size and bit direction will have to slow them down
can be summed up in one word.

ZERO.

Do you really belive that the crypto they are used to cracking from
other nations really restricts itself to nice clean word sizes and
alignments? That the brillance of choosing a 19 bit word size will
stun them into inaction? "Sir, Mr Scott is not following the rules
that we ask the Russians and Chinese to follow in their crypto!"

If they have custom hardware, this is simply a matter of bit re-aligning
on the fly. There is already, in commerical use, reconfigurable hardware
where the design can be redefined _in circuit_ under the control of a
computer. This is perfect to build little 'converter' units to nudge
your 19 bit little endian data into whatever they want, and as fast
as a CPU could want it done.

In my military comm days, we had a custom IO proccessor which could handle
word sizes from 5 upto 32 bits, in either big or little endian, in either
positive or negative logic, with any parity (or even multiple parity per
word). And it could do that for upto 128 high data rate channels at the
same time. All in hardware, all very fast, all built in the mid 70's.

Oh well...

--
Andrew Carol aXcarol@apple.com
(Remove the spam avoiding 'X' from my e-mail address)

Subject: Re: Memo to the Amateur Cipher Designer
Date: Thu, 05 Nov 1998 04:26:33 GMT
From: dscott@networkusa.net
Message-ID: <71r9dp$uv3$1@nnrp1.dejanews.com>
References: <aXcarol-0411981641250001@andrew1.apple.com>
Newsgroups: sci.crypt
Lines: 76

In article <aXcarol-0411981641250001@andrew1.apple.com>,
aXcarol@apple.com (Andrew Carol) wrote:
> In article <71qpr3auf1@nnrp1.dejanews.com>, dscott@networkusa.net wrote:
>
> >I doubt
> >if they have given it as much attention yet. But it is very different
> >from the kind they are use to breaking. Since 19u is built around
> >19 bit boundaries on a PC indian type of machine.
>
> The effect the word size and bit direction will have to slow them down
> can be summed up in one word.
>
> ZERO.
>
> Do you really belive that the crypto they are used to cracking from
> other nations really restricts itself to nice clean word sizes and
> alignments? That the brillance of choosing a 19 bit word size will
> stun them into inaction? "Sir, Mr Scott is not following the rules
> that we ask the Russians and Chinese to follow in their crypto!"
>

I doubt if they can break my 16bit version. But the point is
if it is not ordinary they would have a hard time deteriming the

bit size that was used unless they knew for sure it was a given
size. It just makes there work harder.
  I am not trying to say it is safe becasue it is 19 bits I think
the 16bit version safe. But I still feel that having a non mulitple
of 8 makes it even safer.

> If they have custom hardware, this is simply a matter of bit re-aligning
> on the fly. There is already, in commerical use, reconfigurable hardware
> where the design can be redefined _in circuit_ under the control of a
> computer. This is perfect to build little 'converter' units to nudge
> your 19 bit little endian data into whatever they want, and as fast
> as a CPU could want it done.
>

  What this means is they could use my method for there own
use since my method is not limited like some of the older methods.
Maybe the NSA will use my methods. But if they do they will claim
they invented it first and will change the name.

> In my military comm days, we had a custom IO proccessor which could handle
> word sizes from 5 upto 32 bits, in either big or little endian, in either
> positive or negative logic, with any parity (or even multiple parity per
> word). And it could do that for upto 128 high data rate channels at the
> same time. All in hardware, all very fast, all built in the mid 70's.
>
> Oh well...
>

  So what engineer hasn't worked for them big deal. I liked the 36 bit
one's compliment machine the best. The computer people today
have no concept of what the real machines in the past could do.
I guess you worked for a smaller branch if your stuff only good
t0 32 bits. I am talking about the mid 70's too.
 What I really miss and am not sure why it never caught
on in current microprocessors was the fix point arithmic
fractions scaled -1 to 1 so every thing a fraction. C is
written as if all numbers integer or floating point the
first machines I used where integer or fixed point fraction
Do todays computer jocks even know what the hell we are
talking about.

> --
> Andrew Carol aXcarol@apple.com
> (Remove the spam avoiding 'X' from my e-mail address)
>

--
http://cryptography.org/cgi-bin/crypto.cgi/Misc/scott19u.zip
http://members.xoom.com/ecil/index.htm

----------== Posted via Deja News, The Discussion Network ==----------
http://www.dejanews.com/ Search, Read, Discuss, or Start Your Own

---

Subject: Re: Memo to the Amateur Cipher Designer
Date: Thu, 05 Nov 1998 10:18:43 -0800
From: aXcarol@apple.com (Andrew Carol)
Message-ID: <aXcarol-0511981018450001@andrew1.apple.com>
References: <71r9dp$uv3$1@nnrp1.dejanews.com>
Newsgroups: sci.crypt
Lines: 40

In article <71r9dp$uv3$1@nnrp1.dejanews.com>, dscott@networkusa.net wrote:
> So what engineer hasn't worked for them big deal. I liked the 36 bit
>one's compliment machine the best.

One's complement is, in my humble opinion, a real waste.

Two's complement is much more elegant because the sign is an ordinary
data bit. You can do signed and unsigned arithmetic using exactly the
same hardware. The sign is simply iterpreted as such by the user. With
one's complement, the sign bit is always the sign bit which means your
range of legal unsigned values is cut in half. It's also much harder to
do multiple precision math in one's complement.

Much less flexible. That's why nobody uses it anymore. It's dead Jim.

>I guess you worked for a smaller branch if your stuff only good
>t0 32 bits.

If you consider NORADs air sovereignty mission to be a smaller branch.
Not much data to proccess, only every fixed military air defense radar
in the country to think about (in real time).

Each ROCC system had a 1's complement dual-CPU mainframe, supported by 4
two's complement minicomputers. With lots of custom proccessing hardware
in support. This ran to a room of large vector radar consoles. There are
7 such ROCC systems for North America which all feed NORAD and their stuff.

>Do todays computer jocks even know what the hell we are
>talking about.

My first system, designed mid 50's, had two 2400 bps modems. It was 3

large 19" rack mounted drawers using mechanical resonators for the tone
detection and discrete transisters for everything else. 2400bps did not
reach consumer prices till the early 80's.

Oh well...

--
Andrew Carol aXcarol@apple.com
(Remove the spam avoiding 'X' from my e-mail address)

Subject: Re: Memo to the Amateur Cipher Designer
Date: Thu, 05 Nov 1998 23:50:39 GMT
From: dscott@networkusa.net
Message-ID: <71tdkfovs1@nnrp1.dejanews.com>
References: <aXcarol-0511981018450001@andrew1.apple.com>
Newsgroups: sci.crypt
Lines: 30

In article <aXcarol-0511981018450001@andrew1.apple.com>,
  aXcarol@apple.com (Andrew Carol) wrote:
> In article <71r9dp$uv3$1@nnrp1.dejanews.com>, dscott@networkusa.net wrote:
> > So what engineer hasn't worked for them big deal. I liked the 36 bit
> >one's compliment machine the best.
>
> One's complement is, in my humble opinion, a real waste.
>
> Two's complement is much more elegant because the sign is an ordinary
> data bit. You can do signed and unsigned arithmetic using exactly the
> same hardware. The sign is simply iterpreted as such by the user. With
> one's complement, the sign bit is always the sign bit which means your
> range of legal unsigned values is cut in half. It's also much harder to
> do multiple precision math in one's complement.
>
> Much less flexible. That's why nobody uses it anymore. It's dead Jim.
>
>

  Depends on you definition of flexable. At least the 1's complement
had the same range of numbers in the positive and negative direction
the 2's compliment had an unbalanced range in the sense when you take
absolute values the negativies could be bigger. Also the 2 zeros
was very handy.
--
http://cryptography.org/cgi-bin/crypto.cgi/Misc/scott19u.zip
http://members.xoom.com/ecil/index.htm

-----------== Posted via Deja News, The Discussion Network ==----------
http://www.dejanews.com/ Search, Read, Discuss, or Start Your Own

---

Subject: Re: Memo to the Amateur Cipher Designer
Date: Thu, 05 Nov 1998 17:07:37 -0800
From: aXcarol@apple.com (Andrew Carol)
Message-ID: <aXcarol-0511981707370001@andrew1.apple.com>
References: <71tdkfovs1@nnrp1.dejanews.com>
Newsgroups: sci.crypt
Lines: 22

In article <71tdkfovs1@nnrp1.dejanews.com>, dscott@networkusa.net wrote:

> Depends on you definition of flexable. At least the 1's complement
>had the same range of numbers in the positive and negative direction
>the 2's compliment had an unbalanced range in the sense when you take
>absolute values the negativies could be bigger. Also the 2 zeros
>was very handy.

Very true, but since it's cheaper to implement 2's complement in hardware
(because signed and unsigned use the same hardware), and more
people need to to extended precision math than need 2 zeros, it's what
people use.

I have used both extensively, it's just that I end up using
features of 2's complement almost everyday, and can't think of the
last time I wished I was using the 1's complement.

Oh well...

--
Andrew Carol aXcarol@apple.com
(Remove the spam avoiding 'X' from my e-mail address)

Subject: Re: Memo to the Amateur Cipher Designer
Date: Thu, 5 Nov 1998 18:13:44 -0600
From: "R H Braddam" <rbraddam@aic-fl.com>
Message-ID: <71tf04$g2e$1@server.cntfl.com>
References: <aXcarol-0511981018450001@andrew1.apple.com>
Newsgroups: talk.politics.crypto,comp.security.pgp.discuss,sci.crypt
Lines: 43

Andrew Carol wrote in message ...
>In article <71r9dp$uv3$1@nnrp1.dejanews.com>, >

>My first system, designed mid 50's, had two 2400 bps modems. It was 3
>large 19" rack mounted drawers using mechanical resonators for the tone
>detection and discrete transistors for everything else. 2400bps did not
>reach consumer prices till the early 80's.
>

Your post reminds me of the AN/FST-2B data processor. It took analog input from a search radar and a height-finder radar, digitized the radar returns, and sent the digital data to vector displays (RAndom Plan Position Indicators - RAPPI) and to 2400 bps modems. From there the data went to the AN/FSQ-7 computers of the Air Defense Command. This was in 1967. The T-2 had hundreds of vacuum tubes (12ax7s mostly, if I remember correctly). It had an advanced feature, for the time, of a transistorized Selective Identification Feature which received a code transmitted from an aircraft (or missile) which identified it as friend (or foe if radar tracked an aircraft which did not transmit the code). I guess the facts that there were few modems, and the data was digital, provided the necessary security for data that was not encrypted. Things have changed some since then, and the times, they are a-changing....still.

And still, some things do not change. We have always had, and will always have, *some* well-meaning people who seek jobs in government. Then when they get them, they believe they have the duty and obligation to determine how the rest of us should live. If a little thing like (in the U.S.) the Constitution gets in the way, ignore it. The U.S. Constitution says that U.S. citizens have the right to speak freely, and the right to privacy. In today's modern world with electronic communication by computer, strong encryption is the only way privacy and free speech can be ensured during communication by computer. However, privacy and free speech can not be ensured if even the *threat* of eavesdropping exists. If keys are escrowed with *any* agency, government or commercial, the threat of eavesdropping will exist. There can be no key escrow. Period. You don't take away the rights, directly or indirectly, of the general population to keep criminals from committing crimes. Sure, make it a felony to use cryptography in the commission of a felony. That will help. But depriving the public of the use of cryptography will not prevent criminals from using it.

Subject: Re: Memo to the Amateur Cipher Designer
Date: Thu, 05 Nov 1998 17:13:29 -0800
From: aXcarol@apple.com (Andrew Carol)
Message-ID: <aXcarol-0511981713290001@andrew1.apple.com>
References: <71tf04$g2e$1@server.cntfl.com>
Newsgroups: talk.politics.crypto,comp.security.pgp.discuss,sci.crypt
Lines: 24

In article <71tf04$g2e$1@server.cntfl.com>, "R H Braddam"
<rbraddam@aic-fl.com> wrote:

>Your post reminds me of the AN/FST-2B data processor. It took analog input
>from a search radar and a height-finder radar, digitized the radar returns,
>and sent the digital data to vector displays (RAndom Plan Position
>Indicators - RAPPI) and to 2400 bps modems. From there the data went to the
>AN/FSQ-7 computers of the Air Defense Command. This was in 1967. The T-2 had
>hundreds of vacuum tubes (12ax7s mostly, if I remember correctly).

I worked on the follow-on system, FYQ-9? (Can't remember the last digit).
We took dozens of radars and merged them into a single display and put it
on large vector consoles. Lot's of custom hardware to do the signal procc.

A wonderful system to work on. It was the last of the actual large scale,
local board level repair systems in the Air Force. It was a fairly large
computer system implemented in TTL. Very nice.

All I/O was encrypted (KG-84's, etc). They were much more compact than
the KY-3's and KG-13's my prior system used.

--
Andrew Carol aXcarol@apple.com
(Remove the spam avoiding 'X' from my e-mail address)

Subject: Re: Memo to the Amateur Cipher Designer
Date: Fri, 06 Nov 1998 04:42:58 GMT
From: "Douglas A. Gwyn" <DAGwyn@null.net>
Message-ID: <36427E4F.23A27E06@null.net>
References: <71tf04$g2e$1@server.cntfl.com>
Newsgroups: talk.politics.crypto,sci.crypt
Lines: 28

R H Braddam wrote:
> ... The U.S. Constitution says that
> U.S. citizens have the right to speak freely, and the right to privacy.

The right to privacy is not explicitly spelled out, although
the right to be secure in their homes and possessions is.
The founders of the US generally believed in natural rights
(rights inherent in a person by virtue of his existence)
and "reserved" for the citizens all rights not explicitly

guaranteed by wording in the Constitution. Unfortunately, as predicted by some during the debate over the Bill of Rights, the fact that some rights are explicitly enumerated has led to confusion, such that many people believe that rights are *conveyed* explicitly by the Bill of Rights and that unenumerated rights don't exist or at least are not protected.

> ... But depriving the public of the use
> of cryptography will not prevent criminals from using it.

Yup, and as I previously pointed out, the criminals can use secure unescrowed encryption *within* any key-escrow scheme.

As with most similar laws, notably gun control laws, there is adverse impact on the good guys and little if any good impact on the bad guys. It's easier for politicians to enact such misguided laws and claim to be "doing something" than for them to justify upholding individual freedom.

Subject: Re: Memo to the Amateur Cipher Designer
Date: Fri, 06 Nov 1998 14:38:50 -0800
From: David Sternlight <david@sternlight.com>
Message-ID: <36437A79.23158BFC@sternlight.com>
References: <jgfunj-0611981226220001@dialup104.itexas.net>
<71u125$lm6$1@news.cudenver.edu>
<71tf04$g2e$1@server.cntfl.com>
Newsgroups: talk.politics.crypto,comp.security.pgp.discuss,sci.crypt
Lines: 36

W T Shaw wrote:

> In article <71u125$lm6$1@news.cudenver.edu>, Zero@ouray.cudenver.edu
> (Zero) wrote:
>
> > R H Braddam (rbraddam@aic-fl.com) wrote:
> > : Constitution gets in the way, ignore it. The U.S. Constitution says that
> > : U.S. citizens have the right to speak freely, and the right to privacy. In
> >
> > Not that I disagree with the general content of your post, but to
> > the best of my knowledge, the U.S. Constitution does not even mention
> > privacy.
> >
> The Supremes seem to see that it is implied...so do lots of common folk.
> If you don't, you are merely a sophist.

We've had this discussion ad nauseam here. The Supremes have found a right to certain specific privacies on a case-by-case basis, in the "penumbra" of the Constitution. They have not found a general right to privacy even in the penumbra.

Some specific rights to privacy they've found include the marital bed, etc. They have found no right to privacy having to do with crypto. And there is no absolute right to privacy in a number of other areas ordinarily thought to be so--for example the government may compel productions of one's private papers via lawful subpoena despite the Constitutional "right to be secure in one's papers". And one may, of course, be lawfully arrested despite the "right to be secure in one's person".

Even with respect to speech, the right to free speech is limited, as in the case of falsely shouting "Fire!" in a crowded theatre.

David

Subject: Re: Memo to the Amateur Cipher Designer
Date: Wed, 11 Nov 1998 17:15:49 GMT
From: scott@helsbreth.org (Scott Nelson)
Message-ID: <3649c569.4548892@news.inreach.com>
References: <36437A79.23158BFC@sternlight.com>
Newsgroups: talk.politics.crypto,comp.security.pgp.discuss,sci.crypt
Lines: 54

On Fri, 06 Nov 1998 David Sternlight <david@sternlight.com> wrote:
[edit]
> . . .The Supremes have found a right to
>certain specific privacies on a case-by-case basis, in the "penumbra" of the
>Constitution. They have not found a general right to privacy even in the
>penumbra.
>

The Ninth amendment in the Bill of Rights states
"The enumeration in the Constitution, of certain rights,
shall not be construed to deny or disparage others retained
by the people."

This includes the right to privacy.
However, many rights conflict with others.
"Your right to swing your fist stops where my nose begins"

and all that. You're not entitled to yell "fire" in a
crowded theater, not because you don't have the right to
speak, but rather because the other patrons' rights
override your right to speak thus.

This weighing of other concerns against the general right
to privacy is what the supreme court has adjudicated.
The existence of the general right to privacy has never
seriously been questioned.

>Some specific rights to privacy they've found include the marital bed, etc. They
>have found no right to privacy having to do with crypto.

That's inaccurate. National security may override
a citizens' right to privacy, but that's very different
from not having the right. And IIRC, the supreme court
has rejected all claims by the government that
National Security concerns are overriding in the
case of crypto. They haven't even gotten past the
first amendment yet, much less the forth and ninth.

>And there is no absolute
>right to privacy in a number of other areas ordinarily thought to be so--for
>example the government may compel productions of one's private papers via lawful
>subpoena despite the Constitutional "right to be secure in one's papers". And
>one may, of course, be lawfully arrested despite the "right to be secure in one's
>person".
>
The rights are not absolute rights in the sense that they
override everything, but privacy, like all other rights,
can't just be ignored either. There must be an overriding
reason to violate the right to privacy.

DiehardC 1.03 now available via ftp from
ftp://helsbreth.org/pub/helsbret/random
Scott Nelson <scott@helsbreth.org>

Subject: Re: Memo to the Amateur Cipher Designer
Date: Thu, 12 Nov 1998 06:03:04 GMT
From: pstromer@my-dejanews.com
Message-ID: <72dtmo$566$1@nnrp1.dejanews.com>
References: <364A2925.B027031D@sternlight.com>
<3649c569.4548892@news.inreach.com>
Newsgroups: talk.politics.crypto,comp.security.pgp.discuss,sci.crypt
Lines: 51

In article <364A2925.B027031D@sternlight.com>,
david@sternlight.com wrote:

> The discussion was about a right to privacy in the Constitution. Anyone may
> argue, using the Ninth amendment, that anything they like is a "right". It's
> nonsense without a legal basis.

Perhaps the Supreme Court's decision in Griswold v. Connecticut, 381 U.S. 479
(1965) qualifies as a "legal basis?"

Held: The Connecticut statute forbidding use of contraceptives violates the
right of marital privacy which is within the penumbra of specific guarantees
of the Bill of Rights.

A few choice quotes from Justice Douglas' majority opinion, which I'm sure
will delight Mr. Sternlight:

"specific guarantees in the Bill of Rights have penumbras, formed by
emanations from those guarantees that help give them life and substance"

"We recently referred [p*485] in Mapp v. Ohio, 367 U.S. 643, 656, to the
Fourth Amendment as creating a "right to privacy, no less important than any
other right carefully an particularly reserved to the people."

But it's "only" the Supreme Court, not the Constitution.

> The Supreme Court never acknowledged a general right to privacy, so they
> couldn't very well have adjudicated it.

See the above quotes for a spirited refutation of this comment.

> There needs to be no "overiding reason", just due process and a judicially
> approved reasonable expectation that a search warrant will produce the
> objects sought.

Spoken from someone who doesn't understand the process of obtaining a search
warrant. The judge must be capable of making an independent evaluation of
the merits of the application for the warrant. And the standard for a search
warrant is not "reasonable expectation," but it is "probable cause."

Without "probable cause," the "reasonable expectation" standard results in
"fruits of the poisonous tree" and are inadmissible in court. See the
landmark case of Wong Sun v. United States, 371 U.S. 471 (1963).

--

Philip Stromer

Send me email at pstromer@SPAMSUCKShotmail.com (remove the capital letters).

----------== Posted via Deja News, The Discussion Network ==----------
http://www.dejanews.com/ Search, Read, Discuss, or Start Your Own

---

Subject: Re: Memo to the Amateur Cipher Designer
Date: Thu, 12 Nov 1998 03:31:45 -0600
From: "R H Braddam" <rbraddam@aic-fl.com>
Message-ID: <72e9ur$kb4$1@server.cntfl.com>
References: <364A2925.B027031D@sternlight.com>
<3649c569.4548892@news.inreach.com>
Newsgroups: talk.politics.crypto,comp.security.pgp.discuss,sci.crypt
Lines: 87

I went to this Government Printing Office URL and searched on the word
privacy:
http://www.access.gpo.gov/congress/senate/constitution/index.html

The following is the report provided on my search:
"Searching constitution...Your query was: (PRIVACY)
The database contains 769,912 words in 106 documents.
There are no fields in this database.privacy occurs 218 times in 15
documents.
The search found 15 documents. It took less than a second."

The documents returned to me had a total size of 2,822,965 bytes.

The documents I scanned primarily addressed the 1st, 4th, 9th, and 14th
amendments. This seems to indicate to me that the Congress and the Supreme
Court consider the amendments to be part of the Constitution. So do I. There
are also federal laws addressing privacy.

There is the Privacy Act of 1974, the Privacy Protection Act of 1980, the
Electronic Communications Privacy Act of 1986, and the Computer Matching and
Privacy Protection Act of 1988.

These are just two quotes from Supreme Court justices:
"Justice Douglas, writing the opinion of the Court, asserted that the
``specific guarantees in the Bill of Rights have penumbras, formed by
emanations from those guarantees that help give them life and
substance.''\6\ Thus, while privacy is nowhere mentioned, it is one of the
values served and protected by the First Amendment, through its protection
of associational rights, and by the Third, the Fourth, and the Fifth
Amendments as well. The Justice recurred to the text of the Ninth Amendment,
apparently to support the thought that these penumbral rights are protected
by one Amendment or a complex of Amendments despite the absence of a
specific reference. Justice Goldberg, concurring, devoted several pages to
the Amendment."

"United States v. Padilla, 508 U.S. 77 (1993) (only persons whose privacy or
property interests are violated may object to a search on Fourth Amendment
grounds;"

Or go here, and you can search for anything about anything that's published
by the Government Printing Office. Of course, not everything that's been
published has been transcribed to on-line form, but they say they are
continually updating their databases.
http://www.access.gpo.gov/su_docs/dbsearch.html

I hope this ends the discussion about the right to privacy -- whether it
exists or not. It does, not just in my opinion, but in the opinion of the
Congress and the Supreme Court.

Anonymity is another matter. In 1962 the Supreme Court made a ruling that
indicated that anonymity is a fundamental requirement of free speech. Check
the link below for the full text (200+kb) from which the following excerpt
was taken:
http://cpsr.org/cpsr/free_speech/talley_v_california.txt
(The case was about a requirement for leaflets to have identification
information printed on them.)
"Even the Federalist Papers, written in favor of the adoption of our
Constitution, were published under fictitious names. It is plain that
anonymity has sometimes been assumed for the most constructive purposes.
We have recently had occasion to hold in two cases that there are
times and circumstances when States may not compel members of groups engaged
in the dissemination of ideas to be publicly identified.
Bates v. Little Rock, 361 U.S. 516; N. A. A. C. P. v. Alabama, 357 U.S. 449,
462. The reason for those holdings was that identification and fear of
reprisal might deter perfectly peaceful discussions of public matters of
importance. This broad Los Angeles ordinance is subject to the same
infirmity. We hold that it, like the Griffin, Georgia, ordinance, is void
on its face.
The judgment of the Appellate Department of the Superior Court of
the State of California is reversed and the cause is remanded to it
for further proceedings not inconsistent with this opinion."

You don't have to be a lawyer to do an Internet search.

Now explain to me how we can have anonymity over the Internet without
encryption. Don't bother telling me that this posting is not like a leaflet.
IMHO, that is exactly what it is, in electronic form. If I were afraid of

government retaliation, I would have to encrypt this -- assuming encryption was available without escrowed keys, as it is now. If we want to keep the capability we have now, and improve upon it, we have to take action. Try http://www.ciec.org/ and send email urging action to block the Embassy chip. One message might help, two would be better, and if everyone reading this responds you might get a snowball rolling that will prevent the Embassy from even getting started.

Subject: Re: Memo to the Amateur Cipher Designer
Date: 18 Nov 1998 03:11:25 GMT
From: lamontg@bite.me.spammers
Message-ID: <72tdst$1ieu$1@nntp6.u.washington.edu>
References: <72e9ur$kb4$1@server.cntfl.com>
Newsgroups: talk.politics.crypto,comp.security.pgp.discuss,sci.crypt
Lines: 62

"R H Braddam" <rbraddam@aic-fl.com> writes:
>The documents I scanned primarily addressed the 1st, 4th, 9th, and 14th
>amendments. This seems to indicate to me that the Congress and the Supreme
>Court consider the amendments to be part of the Constitution. So do I. There
>are also federal laws addressing privacy.

There is no explicit right to privacy in the Constitution of the United States. From Roe v. Wade:

``The Constitution does not explicitly mention any right of privacy. In a line of decisions, however, going back perhaps as far as Union Pacific R. Co. v. Botsford, 141 U.S. 250, 251 (1891), the Court has recognized that a right of personal privacy, or a guarantee of certain areas or zones of privacy, does exist under the Constitution. In varying contexts, the Court or individual Justices have, indeed, found at least the roots of that right in the First Amendment, Stanley v. Georgia, 394 U.S. 557 , 564 (1969); in the Fourth and Fifth Amendments, Terry v. Ohio, 392 U.S. 1 , 8-9 (1968), Katz v. United States, 389 U.S. 347 , 350 (1967), Boyd v. United States, 116 U.S. 616 (1886), see Olmstead v. United States, 277 U.S. 438 , 478 (1928) (Brandeis, J., dissenting); in the penumbras of the Bill of Rights, Griswold v. Connecticut, 381 U.S. at 484-485 ; in the Ninth Amendment, id. at 486 (Goldberg, J., concurring); or in the concept of liberty guaranteed by the first section of the Fourteenth Amendment, see Meyer v. Nebraska, 262 U.S. 390 , 399 (1923). These decisions make it clear that only personal rights that can be deemed "fundamental" or "implicit in the concept of ordered liberty," Palko v. Connecticut, 302 U.S. 319, 325 (1937), are included in this guarantee of personal privacy. They also make it clear that the right has some extension to activities relating to marriage, Loving v. Virginia, 388 U.S. 1 , 12 (1967); procreation, Skinner v. Oklahoma, 316 U.S. 535 , 541-542 (1942); contraception, Eisenstadt v. Baird, 405 U.S. at 453-454; id. at 460, 463-465 [p*153] (WHITE, J., concurring in result); family relationships, Prince v. Massachusetts, 321 U.S. 158, 166 (1944); and childrearing and education, Pierce v. Society of Sisters, 268 U.S. 510 , 535 (1925), Meyer v. Nebraska, supra.

``This right of privacy, whether it be founded in the Fourteenth Amendment's concept of personal liberty and restrictions upon state action, as we feel it is, or, as the District Court determined, in the Ninth Amendment's reservation of rights to the people, is broad enough to encompass a woman's decision whether or not to terminate her pregnancy.''

>I hope this ends the discussion about the right to privacy -- whether it
>exists or not. It does, not just in my opinion, but in the opinion of the
>Congress and the Supreme Court.

It has never, however, been an explicit part of the Constitition. It has merely been argued that it can be based on the Constitution. And I do view this as "merely", since it is possible that a more conservative court could argue against it and weaken it.

>Anonymity is another matter. In 1962 the Supreme Court made a ruling that
>indicated that anonymity is a fundamental requirement of free speech.
[...]
>Now explain to me how we can have anonymity over the Internet without
>encryption.

Anonymity is easy on the Internet. Just forge all your articles. No Crypto required.

--
Lamont Granquist (lamontg@u.washington.edu)
ICBM: 47 39'23"N 122 18'19"W

Subject: Re: Memo to the Amateur Cipher Designer
Date: 6 Nov 1998 09:10:16 GMT
From: olson@umbc.edu (Bryan G. Olson; CMSC (G))
Message-ID: <71uedo$hv8$2@news.umbc.edu>
References: <aXcarol-0511981018450001@andrew1.apple.com>
Newsgroups: sci.crypt
Lines: 25

Andrew Carol (aXcarol@apple.com) wrote:

: One's complement is, in my humble opinion, a real waste.

: Two's complement is much more elegant because the sign is an ordinary
: data bit. You can do signed and unsigned arithmetic using exactly the
: same hardware. The sign is simply iterpreted as such by the user. With
: one's complement, the sign bit is always the sign bit which means your
: range of legal unsigned values is cut in half.

Not so. Two's complement is simply arithmetic mod 2^WordSize,
while one's complement is mod (2^WordSize)-1. With either one
we use the same addition and subtraction operations for both
signed and unsigned. We're left with the the modular remainder
and we choose whether to interpret the values with the one
bit set as the least residue or to subtract the modulus from
the least residue.

: It's also much harder to
: do multiple precision math in one's complement.

True. That missing value is a huge pain.

--Bryan

Subject: Re: Memo to the Amateur Cipher Designer
Date: 7 Nov 1998 15:11:15 GMT
From: jsavard@freenet.edmonton.ab.ca ()
Message-ID: <721nuj$57m$1@news.sas.ab.ca>
References: <36427A25.C9CFD81D@null.net>
   <aXcarol-0511981018450001@andrew1.apple.com>
Newsgroups: sci.crypt
Lines: 31

Douglas A. Gwyn (DAGwyn@null.net) wrote:
: Andrew Carol wrote:
: > One's complement is, in my humble opinion, a real waste.
: > Two's complement is much more elegant because the sign is an ordinary
: > data bit. You can do signed and unsigned arithmetic using exactly the
: > same hardware. The sign is simply iterpreted as such by the user. With
: > one's complement, the sign bit is always the sign bit which means your
: > range of legal unsigned values is cut in half. It's also much harder to
: > do multiple precision math in one's complement.

: You're completely wrong. Operations using ones-complement and
: twos-complement representation are very similar; CDC chose to
: use ones-complement because it was slightly faster (negation
: doesn't require any carry cycles).

Well, it is true that less circuitry is required to add a negative integer
to a positive integer in two's complement; one does need extra gates to
test for opposing signs and adjust the result in one's complement.

Two's complement is the reasonable representation for integers.

But for floating-point numbers, composed of separate fields, the normal
representation is sign-magnitude for the mantissa, and excess-N for the
exponent. If one wants one's floating-point numbers to compare using
integer comparison operations, then inverting the other bits of the number
when the sign is negative - a one's complement on the mantissa, but the
exponent is also inverted - makes sense, because field boundaries are not
straddled. (Of course, the architecture should be big-endian, so that
integer compares work on strings too, if you're doing this...)

John Savard

Subject: Re: Memo to the Amateur Cipher Designer
Date: Sun, 08 Nov 1998 06:51:57 GMT
From: "Douglas A. Gwyn" <DAGwyn@null.net>
Message-ID: <36453F84.9E2E6827@null.net>
References: <721nuj$57m$1@news.sas.ab.ca>
Newsgroups: sci.crypt
Lines: 18

jsavard@freenet.edmonton.ab.ca wrote:
> Well, it is true that less circuitry is required to add a negative integer
> to a positive integer in two's complement; one does need extra gates to
> test for opposing signs and adjust the result in one's complement.
> Two's complement is the reasonable representation for integers.

Which representation produces faster results depends on the
exact mix of operations. I've programmed both, and either
representation is reasonable for most purposes.

> ... (Of course, the architecture should be big-endian, so that
> integer compares work on strings too, if you're doing this...)

Endianness is another architectural property that has pros
and cons on both sides.

Anyway, this is outside the sci.crypt charter; I just
couldn't stand by while ones-complement was being badmouthed.

Subject: Re: Memo to the Amateur Cipher Designer
Date: Wed, 04 Nov 1998 19:48:04 -0600
From: jgfunj@EnqvbSerrGrknf.pbz (W T Shaw)
Message-ID: <jgfunj-0411981948040001@dialup150.itexas.net>
References: <71qpr3auf1@nnrp1.dejanews.com>
Newsgroups: sci.crypt
Lines: 103

In article <71qpr3auf1@nnrp1.dejanews.com>, dscott@networkusa.net wrote:
>
> If the NSA is doing its job at all any encryption that is used at
> all on the net would be analyzed by them. I am sure megabucks where
> spent on PGP since it is so common. I think my stuff is stronger
> than IDEA or the FISHY methods but since not in public eye I doubt
> if they have given it as much attention yet. But it is very different
> from the kind they are use to breaking. Since 19u is built around
> 19 bit boundaries on a PC indian type of machine. The program has
> to be decrypt by several passes in the reverse direction. And as
> the R of RSA is now pushing or talking about. It is a true all or
> nothing encryption something that may be beyond the B S crypto
> class of peoples current limited mind kind of thinking.
>

I expect they already have looked at your various algorithms. After all it is their job to do this sort of thing. About getting word on what they feel about someones work in particular, that is rare, but it does happen.

We keep hearing how difficult it is to come up with algorithms, so in an hour of thinking and calculating, here is a new one for you, Cipher 4045:

PW5.B40.D8.T8.D4.B22.W3.T12.W4.B32M.D6.T24.D8.B40.CW5 --Cipher "4045" features three tranposition boxes, 8/12/24 equivalent to 15/29/79, or total of 123 bits. The plaintext and ciphertext are both in base 40 and the input/output block size is 40/45 characters. Without increasing the block size, the transposition boxes can be as big as 64/48/72, for equivalent keys of 290/203/345, or 838 bits all together. Even with no transpositions, straight pass through, this cipher in translation mode exhibits some diffusion. In cryptoanalysis, the weakest length is in the second transposition, so part of the keylength can be somewhat divided at that point.

Note: the above formula is fully descriptive of the algorithm. In case you missed the introduction of this system of notation, here is a repeat of the first three algorithms that tends to explain it by example:

PW1.B100.D2.T12.D4.B22.CW3 describes an algorithm I presented at the ACA in Los Angeles in August. CYCLISTE named this algorithm RIMFIRE, since it uses takes a 22 base shot at all familiar ascii characters , almost all of those in the set of 100.

In it plaintext (P) characters are converted individually(W1, words of one character) from a set of 100 elements (B100) to two digits (D2, normal base 10). The digits undergo transposition in blocks of twelve (T12). Digits in groups of four (D4) are converted to base 22 characters (B22) in blocks of three ciphertext letters (CW3).

The keylength is entirely in the possible permutations of the 12 elements in the transposition stage, only 479,001,600 different keys. This cipher is targeted at the twilight above hand breakable ciphers yet somewhat easy to break with a computer.

PW3.B22.D4.GVA6X11PC1.P.D6.B32M.CW4 is the algorithm that I am using with the 6 wheels using alphabets of 11 characters in the FLAT CYLINDER EXAMPLE of the GVA.

In it, plaintext in words of three letters each (PW3) from a base 22 character set (B22) produce groups of 4 digits each (D4). Using the Grandview Algorithm as a build block (GVA), noting its size (6X11), and a selection of a group of six digits (D6) to be changed to base 32, the medium table available, (B32M), producing ciphertext words of four characters each (CW4).

PW5.B40.D8.GVA11X16PC2.B16.CW11 is a slightly more complicated implementation of the Grandview Algorithm called CAPS, because it is worked on a cylinder made from punched plastic milk bottle caps, and is more uniform and compact than the push-pop cylinder.

In it, plaintext words of five characters each (PW5) from a set of 22 (B22) produce groups of 8 digits each (D8). The Grandview Algorithm is exercised on a cylinder of 11 wheels having 16 charcters each with two hexadecimal (B16), containing eleven characters in each ciphertext word (CW11).

This is lots stronger than the flat cylinder example as it has 3375 possible outputs for each input group. It can work from any of the 225 available pathcodes. Using a hex set on the cylinder means that digits are a subset of it, and easily applied. Using the GVA Alternate Form, it can work with 4096 different pathcodes.

PW5.B40.D8.T8.D4.B22.W3.T12.W4.B32M.D6.T24.D8.B40.CW5 --Cipher "4045" features three tranposition boxes, 8/12/24 equivalent to 15/29/79, or total of 123 bits. The plaintext and ciphertext are both in base 40 and

the input/output block size is 40/45 characters. Without increasing the block size, the transposition boxes can be as big as 64/48/72, for equivalent keys of 290/203/345, or 838 bits all together. Even with no transpositions, straight pass through, this cipher in translation mode exhibits some diffusion. The weakest length is in the second transposition.

Note: the above formula is fully descriptive of the algorithm. In case you missed the introduction of this system of notation, here is a repeat of the first three algorithms:
--

The public is harder to steamroller than some might think.

Decrypt with ROT13 to get correct email address.

Subject: Re: Memo to the Amateur Cipher Designer
Date: Fri, 06 Nov 1998 07:51:19 GMT
From: schneier@counterpane.com (Bruce Schneier)
Message-ID: <3649aa3b.2925663@news.visi.com>
References: <jgfunj-0411981159440001@207.22.198.223>
Newsgroups: sci.crypt
Lines: 28

On Wed, 04 Nov 1998 11:59:43 -0600, jgfunj@EnqvbSerrGrknf.pbz (W T Shaw) wrote:
>Cryptography is a special class, probably not entirely alone as other
>subjects might also get similiar attention. Consider what is apt to
>happen: PTO gets some crypto related application; it gets put into a pile
>since it might be messing with ideas that could classified. The
>application is properly forwarded to the Defense Department for review, if
>not directly to NSA itself.

The NSA does not comment on patentability. Actually, I'm pretty sure they don't review crypto patent apps anymore. That kind of thing only worked in the 70s and early 80s.

>It would be on the recommendations of the particular agencies that handled
>the details of inspection that the PTO office would act. The paperwork
>could be simply passed back quickly to PTO and processed as representing
>something trivial, but patentable all the same, returned and rejected to
>the submitter with no reasons given, or held for further study. From
>there, more options could click in. I bet you Ritter's work got lots more
>than a quick glance.

Sounds like a good idea, but the patent office doesn't work that way.

Bruce
****
Bruce Schneier, President, Counterpane Systems Phone: 612-823-1098
101 E Minnehaha Parkway, Minneapolis, MN 55419 Fax: 612-823-1590
Free crypto newsletter. See: http://www.counterpane.com

Subject: Re: Memo to the Amateur Cipher Designer
Date: Fri, 06 Nov 1998 12:47:00 -0600
From: jgfunj@EnqvbSerrGrknf.pbz (W T Shaw)
Message-ID: <jgfunj-0611981247000001@dialup104.itexas.net>
References: <3649aa3b.2925663@news.visi.com>
Newsgroups: sci.crypt
Lines: 17

In article <3649aa3b.2925663@news.visi.com>, schneier@counterpane.com (Bruce Schneier) wrote:
>
> The NSA does not comment on patentability. Actually, I'm pretty sure
> they don't review crypto patent apps anymore. That kind of thing only
> worked in the 70s and early 80s.
....
Don't be so sure. Sent in a technically-heavy sounding fuzzy description for a megathermoneutrogismo with 10 pages of obscure drawings and be sure to get somebody's attention. I don't expect NSA to be incompetent, which means it should it sops up and funnels ALL the easy leads to the company store for analysis.
--

The public is harder to steamroller than some might think.

Decrypt with ROT13 to get correct email address.

Subject: Re: Memo to the Amateur Cipher Designer
Date: Tue, 10 Nov 1998 03:33:58 GMT
From: ritter@io.com (Terry Ritter)
Message-ID: <3647b41e.4433230@news.io.com>
References: <jgfunj-0411981159440001@207.22.198.223>
Newsgroups: sci.crypt
Lines: 59

On Wed, 04 Nov 1998 11:59:43 -0600, in
<jgfunj-0411981159440001@207.22.198.223>, in sci.crypt

jgfunj@EnqvbSerrGrknf.pbz (W T Shaw) wrote:

>[...]
>Cryptography is a special class, probably not entirely alone as other
>subjects might also get similiar attention. Consider what is apt to
>happen: PTO gets some crypto related application; it gets put into a pile
>since it might be messing with ideas that could classified. The
>application is properly forwarded to the Defense Department for review, if
>not directly to NSA itself.

It is my understanding that there is "a NSA desk" in the PTO which
does review crypto applications. If so, it must be a tough job.

There is a whole section to the MPEP (Manual of Patent Examination
Procedure) that deals with secrecy orders, and anybody can read that.
It seem unlikely, however, that it could apply to a patent whose
information had been publicly revealed. Not only would a secrecy
order be ineffective in that case, it would actually highlight a
significant idea, so that would not be done.

>It would be on the recommendations of the particular agencies that handled
>the details of inspection that the PTO office would act. The paperwork
>could be simply passed back quickly to PTO and processed as representing
>something trivial, but patentable all the same, returned and rejected to
>the submitter with no reasons given, or held for further study. From
>there, more options could click in. I bet you Ritter's work got lots more
>than a quick glance.

My Dynamic Substitution patent file actually *disappeared* from the
PTO for a while in 1990. Presumably it was signed out, but the file
was physically out of the PTO. This was *after* allowance by the
examiner, but before publication. After repeated telephone requests
for information, eventually I was passed to the head of the
publication department, who ordered the file *recalled* from wherever
it was, although she would not say where that was, or why it was
there. Very spooky in 1990, but perhaps more humorous in retrospect.
The information already had been *published* on sci.crypt and was
ready for print in Cryptologia. But only after the file was ordered
recalled could it be assigned a publication slot and an issue date,
and so eventually issue.

One might speculate that there was *some* sort of a review going on,
but certainly not for patentability. There were fewer crypto patents
in those days, so there may have been greater anxiety. Still, the
content was already published!

There is no way for me to know whether the patent is being used by the
government. But if someone were to provide the name of a machine
which they claim does use it, I could request a formal inquiry.

Terry Ritter ritter@io.com http://www.io.com/~ritter/
Crypto Glossary http://www.io.com/~ritter/GLOSSARY.HTM

Subject: Re: Memo to the Amateur Cipher Designer
Date: Tue, 10 Nov 1998 14:18:34 GMT
From: malinov@mindless.com
Message-ID: <729hvqehh1@nnrp1.dejanews.com>
References: <3647b41e.4433230@news.io.com>
Newsgroups: sci.crypt
Lines: 27

Terry Ritter said something like . . .

> It is my understanding that there is "a NSA desk" in the PTO which
> does review crypto applications. If so, it must be a tough job.

Not at all, having worked there many years. All applications are reviewed for
subject matter which might involve national security. Applications by US
citizens which involve real cryptography are copied; the copy is sent to
someone at NSA for review. Secrecy orders are rare. They are not easily
impressed.

> My Dynamic Substitution patent file actually *disappeared* from the
> PTO for a while in 1990. Presumably it was signed out, but the file
> was physically out of the PTO.

Welcome to the bureaucracy. I can't say for sure, but I'd bet good money
there was a dumb reason your file wasn't where it should have been.
Government dumbness outweighs conspiracy by several orders of magnitude.

David Cain
Ex-PTO Crypto

--
Power belongs to those who dare . . . Sapere Aude

----------== Posted via Deja News, The Discussion Network ==----------
http://www.dejanews.com/ Search, Read, Discuss, or Start Your Own

---

Subject: Re: Memo to the Amateur Cipher Designer
Date: Tue, 10 Nov 1998 18:43:18 GMT
From: ritter@io.com (Terry Ritter)
Message-ID: <3648892f.4989511@news.io.com>
References: <729hvqehh1@nnrp1.dejanews.com>
Newsgroups: sci.crypt
Lines: 51

On Tue, 10 Nov 1998 14:18:34 GMT, in
<729hvqehh1@nnrp1.dejanews.com>, in sci.crypt malinov@mindless.com
wrote:

>Terry Ritter said something like . . .
>
>> It is my understanding that there is "a NSA desk" in the PTO which
>> does review crypto applications. If so, it must be a tough job.
>
>Not at all, having worked there many years. All applications are reviewed for
>subject matter which might involve national security. Applications by US
>citizens which involve real cryptography are copied; the copy is sent to
>someone at NSA for review.

You mean they send over the entire input to the crypto section? If
not, the person who makes the decision about what to send over would
seem to need some knowledge beyond patents.

>Secrecy orders are rare. They are not easily
>impressed.

It's more than that: Secrecy orders on crypto inherently identify
systems which the government finds particularly interesting. Few
secrecy orders last forever, so the very use of this power eventually
highlights what it is supposed to hide. There may be very few
situations where that is warranted.

And of course a secrecy order is really only useful on ideas which
have not been published.

>> My Dynamic Substitution patent file actually *disappeared* from the
>> PTO for a while in 1990. Presumably it was signed out, but the file
>> was physically out of the PTO.
>
>Welcome to the bureaucracy. I can't say for sure, but I'd bet good money
>there was a dumb reason your file wasn't where it should have been.
>Government dumbness outweighs conspiracy by several orders of magnitude.

Normally very true, although in this case the clerks were more
hush-hush close-mouthed than in the usual screw-up. I was eventually
bucked up to the department head Herself and she wouldn't tell me
where the file was or why, which seems unusual to this day.

Terry Ritter ritter@io.com http://www.io.com/~ritter/
Crypto Glossary http://www.io.com/~ritter/GLOSSARY.HTM

Subject: Re: Memo to the Amateur Cipher Designer
Date: Wed, 11 Nov 1998 13:54:56 GMT
From: malinov@mindless.com
Message-ID: <72c4vgkau1@nnrp1.dejanews.com>
References: <3648892f.4989511@news.io.com>
Newsgroups: sci.crypt
Lines: 47

Terry said something like . . .

> You mean they send over the entire input to the crypto section? If
> not, the person who makes the decision about what to send over would
> seem to need some knowledge beyond patents.

They send all the nuts and bolts crypto to NSA - i.e. no protocols, no
business apps, just algorithms for turning plaintext into ciphertext. At one
time, half of the screeners were crypto examiners. But it doesn't take a
cryptographer to recognize cryptography. NSA decides if there's anything
worth looking at.

> It's more than that: Secrecy orders on crypto inherently identify
> systems which the government finds particularly interesting. Few
> secrecy orders last forever, so the very use of this power eventually
> highlights what it is supposed to hide. There may be very few
> situations where that is warranted.

Actually, I was being glib. Secrecy orders are only imposed on government
owned systems, usually classified from birth. Ostensibly, they're looking for
projects they already own - although I suspect they use the patent review
system to keep an extra eye on developing technology.

Secrecy orders are routinely rescinded when the information they protect is
outed.

With few exceptions, the government does not file patent applications for cryptography. I gave NSA a patent for a key escrow system, and that was probably the only NSA application I ever saw. Never Say Anything.

> Normally very true, although in this case the clerks were more
> hush-hush close-mouthed than in the usual screw-up. I was eventually
> bucked up to the department head Herself and she wouldn't tell me
> where the file was or why, which seems unusual to this day.

Admissions create exposure for a bureaucrat, especially when dealing with the public. Anything is possible, but in fourteen years at the PTO, I witnessed no covert operations and twenty million instances of hush-hush bureaucratic bumbling. My money is still on "whoops, goof, what application?"

David Cain

--
Power belongs to those who dare . . . Sapere Aude

----------== Posted via Deja News, The Discussion Network ==----------
http://www.dejanews.com/ Search, Read, Discuss, or Start Your Own

---

Subject: Re: Memo to the Amateur Cipher Designer
Date: Tue, 10 Nov 1998 19:42:53 GMT
From: jsavard@tenMAPSONeerf.edmonton.ab.ca (John Savard)
Message-ID: <36489454.11838628@news.prosurfr.com>
References: <729hvqehh1@nnrp1.dejanews.com>
Newsgroups: sci.crypt
Lines: 31

malinov@mindless.com wrote, in part:

>Secrecy orders are rare. They are not easily
>impressed.

I'm not surprised,

- considering the embarassment resulting from two unwarranted secrecy orders during the 1970s, and

- given that ciphers, and not cryptanalytic approaches, are usually the subjects of patents, while there probably is quite a bit in the field of cryptanalysis that the NSA would like to have remain secret, unless someone comes up with a radically new idea, on the order of public-key cryptography, what is there to classify?

Existing block ciphers include elements that can be strung together on small computers with more complexity; enough so that unbreakable encryption is already a reality for anyone who wants it.

Quantum computing ... that could change the situation, both for ways of making it practical, and for ciphers specifically designed to be resistant to it.

The NSA has many secrets, I'm sure ... but how to make a really, really secure cipher is not one of them. (What things you don't have to bother with, and yet still be secure ... now _that_ is, no doubt, highly classified.)

John Savard
http://www.freenet.edmonton.ab.ca/~jsavard/index.html

Subject: Re: Memo to the Amateur Cipher Designer
Date: Tue, 10 Nov 1998 16:42:05 GMT
From: bo.doemstedt@mbox200.swipnet.se (Bo Dömstedt)
Message-ID: <364a606f.12593692@nntpserver.swip.net>
References: <3647b41e.4433230@news.io.com>
Newsgroups: sci.crypt
Lines: 19

ritter@io.com (Terry Ritter) wrote:
>It is my understanding that there is "a NSA desk" in the PTO which
>does review crypto applications. If so, it must be a tough job.
>
>There is a whole section to the MPEP (Manual of Patent Examination
>Procedure) that deals with secrecy orders, and anybody can read that.
[...]
Fantastic! What would happen if some foreigner, such as me, would file a cipher patent application? Is there some international co-operation of secrecy orders? :)

Bo Dömstedt
Cryptographer
Protego Information AB
Malmoe,Sweden

SG100 Hardware Random Number Generator
http://www.protego.se/sg100_en.htm

Subject: Re: Memo to the Amateur Cipher Designer
Date: Wed, 11 Nov 1998 16:25:00 GMT
From: malinov@mindless.com
Message-ID: <72cdot$s3i$1@nnrp1.dejanews.com>
References: <364a606f.12593692@nntpserver.swip.net>
Newsgroups: sci.crypt
Lines: 22

Bo said something like . . .

> Fantastic! What would happen if some foreigner, such as me, would
> file a cipher patent application? Is there some international
> co-operation of secrecy orders? :)

If you filed an application from Sweden in the US, it would not even be shown to NSA. If you lived in the US, or worked with a US citizen as a joint inventor, they'd take a look. If any of it came from here, it might be ours. But the US is usually more than happy to publish foreign secrets. ;)

The US has an agreement with NATO on secrecy orders, but that just allows NATO members a mechanism to file classified applications in the US. All original classifiction would be done in the home country.

David Cain

--
Power belongs to those who dare . . . Sapere Aude

----------== Posted via Deja News, The Discussion Network ==----------
http://www.dejanews.com/ Search, Read, Discuss, or Start Your Own

---

Subject: Re: Memo to the Amateur Cipher Designer
Date: Fri, 30 Oct 1998 20:01:08 GMT
From: ritter@io.com (Terry Ritter)
Message-ID: <363a1ada.12425654@news.io.com>
References: <3639d8bf.741333@news.visi.com>
Newsgroups: sci.crypt
Lines: 88

On Fri, 30 Oct 1998 15:31:06 GMT, in <3639d8bf.741333@news.visi.com>,
in sci.crypt schneier@counterpane.com (Bruce Schneier) wrote:

>[...]
>Some cryptographic algorithms are patented, yes. I'm not sure how
>that's relevent.

For one thing, the patent literature is far more "archival" than almost any academic journal.

>I do not know of any academic cryptographers that
>regularly look through the U.S. patent system.

I have no doubt that understanding the patent literature is substantially more difficult than understanding academic papers. Still, it *is* published technology. Failing to know published technology means failing to be an expert in the field.

>Patents are not a
>peer-reviewed publication. If an academic (in any discipline)
>presented a list of patents to his tenure review board, they would not
>be considered publication.

Obviously we need to adjust our understanding of an academic as an expert. People can hardly be experts on a field if they refuse to keep informed of parts of it.

The implication that any field consists only of that in the academic literature is not only insulting, it is also dangerously wrong for academia. When I was doing processor design at Motorola, some of my former professors came over, and simply had no idea how MOS design worked. They were irrelevant, and now we see how that happens. (Of course, Motorola could not have had a public web page on our design technology, but the profs probably wouldn't have used it anyway. They were arrogant.)

>Patents are not relevent to academic publication. There are academics
>who also patent. Ritter is an example of someone who does not publish
>(in an academic sense) but does patent. His writings are generally
>ignored by the academic community. I'm sorry this is true; I'd like
>it to be different.

It's *not* true that I don't publish; I just don't publish in a particular form and outlet.

It *is* true that I rarely even consider a magazine publication anymore, let alone an academic journal. But the material is there. The content is there. If academics fail to click up my pages and read them, I probably can't force them to do so.

>If I wrote a paper on Ritters designs, citing his patents and Usenet
>postings and webpages, I believe that I would have a lot of trouble
>getting it published.

Now, I think *that* is a serious problem: If your paper is reviewed not by its content, but by where the original ideas came from, or what its references are, we are back to having a serious conflict with the meaning of Science.

And I have been a peer reviewer: I claim that publication references should be checked for correctness and applicability to the content; that's it. Authors who allow reviewers to remove valid references risk real damage to their reputation when prior publications come to light. Acknowledging the prior art is not optional. The appearance of having used someone else's work for your advantage is not going to help people trust you with their own newest work.

>[...]
>I'm really trying to help here. I am not being inconsiderate to the
>program committee of FSE. I am not making use of the FSE name to
>argue my own position. I don't have a position. I have more than my
>share of ad hominum arguments on sci.crypt, and I would appreciate a
>little bit of curtesy.

Everybody has a bad day once in a while (or more often for me). We just need to tolerate stuff sometimes.

Terry Ritter ritter@io.com http://www.io.com/~ritter/
Crypto Glossary http://www.io.com/~ritter/GLOSSARY.HTM

Subject: Re: Memo to the Amateur Cipher Designer
Date: Fri, 30 Oct 1998 20:34:52 GMT
From: schneier@counterpane.com (Bruce Schneier)
Message-ID: <363a2037.500639@news.visi.com>
References: <363a1ada.12425654@news.io.com>
Newsgroups: sci.crypt
Lines: 133

On Fri, 30 Oct 1998 20:01:08 GMT, ritter@io.com (Terry Ritter) wrote:

>
>On Fri, 30 Oct 1998 15:31:06 GMT, in <3639d8bf.741333@news.visi.com>,
>in sci.crypt schneier@counterpane.com (Bruce Schneier) wrote:
>
>>[...]
>>Some cryptographic algorithms are patented, yes. I'm not sure how
>>that's relevent.
>
>For one thing, the patent literature is far more "archival" than
>almost any academic journal.
>
>>I do not know of any academic cryptographers that
>>regularly look through the U.S. patent system.
>
>I have no doubt that understanding the patent literature is
>substantially more difficult than understanding academic papers.
>Still, it *is* published technology. Failing to know published
>technology means failing to be an expert in the field.

Look, I'm not arguing about whether this is good or beneficial or useful. I am just stating some facts. Perhaps it is true that someone who does not regularly read the patent output stream should not be considered an expert in the field. I don't know. None of this is related to my point, which is about what actually does happen.

>>Patents are not a
>>peer-reviewed publication. If an academic (in any discipline)
>>presented a list of patents to his tenure review board, they would not
>>be considered publication.

>Obviously we need to adjust our understanding of an academic as an
>expert. People can hardly be experts on a field if they refuse to
>keep informed of parts of it.

Adjust away. That's fine.

>The implication that any field consists only of that in the academic
>literature is not only insulting, it is also dangerously wrong for
>academia. When I was doing processor design at Motorola, some of my
>former professors came over, and simply had no idea how MOS design
>worked. They were irrelevant, and now we see how that happens. (Of
>course, Motorola could not have had a public web page on our design
>technology, but the profs probably wouldn't have used it anyway. They
>were arrogant.)

I don't mean to insult you. I am just the bearer of news. You are free to be insulted. Maybe you are correct. I have found that it is better to play by the rules as they are than it is to complain about how unfair they are. I have found this to be true, even if the rules

are unfair.

>>Patents are not relevent to academic publication. There are academics
>>who also patent. Ritter is an example of someone who does not publish
>>(in an academic sense) but does patent. His writings are generally
>>ignored by the academic community. I'm sorry this is true; I'd like
>>it to be different.
>
>It's *not* true that I don't publish; I just don't publish in a
>particular form and outlet.

Which is why I added the parenthetical remark above.

>It *is* true that I rarely even consider a magazine publication
>anymore, let alone an academic journal. But the material is there.
>The content is there. If academics fail to click up my pages and read
>them, I probably can't force them to do so.

You cannot force them to do so.

>>If I wrote a paper on Ritters designs, citing his patents and Usenet
>>postings and webpages, I believe that I would have a lot of trouble
>>getting it published.
>
>Now, I think *that* is a serious problem: If your paper is reviewed
>not by its content, but by where the original ideas came from, or what
>its references are, we are back to having a serious conflict with the
>meaning of Science.

Possibly we do. But I don't have any conflict here. I am simply
stating what I believe to be the realities of publishing in journals,
conferences, or workshops. I would prefer it if I could get such a
paper published.

A few weeks ago I mentioned something that happened to me in one of my
papers. In the submitted version of one of the two papers on
related-key cryptanalysis, we talked about the analysis of S1, a
cipher posted to sci-crypt and believed by some to be Skipjack. (As
it turned out, it wasn't.) Among the anonymous review comments we
received was a note saying that ciphers posted to sci.crypt are not
considered interesting (I forget the exact wording), and that we
should delete that section from our paper. We did.

Look, I didn't write the comment. I would have liked to include our
result (well David Wagner's result) on S1 in the paper. I thought it
was unfair. But it did happen. This is a true story.

>And I have been a peer reviewer: I claim that publication references
>should be checked for correctness and applicability to the content;
>that's it. Authors who allow reviewers to remove valid references
>risk real damage to their reputation when prior publications come to
>light. Acknowledging the prior art is not optional. The appearance
>of having used someone else's work for your advantage is not going to
>help people trust you with their own newest work.

I've never heard of reviewers removing references. I've heard of
reviewers asking for changes in content. And remember, there is an
agreement on both sides here. The author is agreeing to make the
change in order for the paper to be published. We willingly removed
the section on S1 from our paper because we wanted it to be published,
and didn't feel that we lost much in removing it.

>>[...]
>>I'm really trying to help here. I am not being inconsiderate to the
>>program committee of FSE. I am not making use of the FSE name to
>>argue my own position. I don't have a position. I have more than my
>>share of ad hominum arguments on sci.crypt, and I would appreciate a
>>little bit of curtesy.
>
>Everybody has a bad day once in a while (or more often for me). We
>just need to tolerate stuff sometimes.

I'm about as tolerant as you can get, but the personal attacks on me
have been going on for a lot longer than a day and are getting worse.
I have just installed a newsreader with a killfile, though, and things
are looking up.

Bruce
****
Bruce Schneier, President, Counterpane Systems Phone: 612-823-1098
101 E Minnehaha Parkway, Minneapolis, MN 55419 Fax: 612-823-1590
Free crypto newsletter. See: http://www.counterpane.com

Subject: Re: Memo to the Amateur Cipher Designer
Date: Sat, 31 Oct 1998 00:13:17 GMT
From: dscott@networkusa.net
Message-ID: <71dkmt$763$1@nnrp1.dejanews.com>
References: <363a2037.500639@news.visi.com>
Newsgroups: sci.crypt
Lines: 26

In article <363a2037.500639@news.visi.com>,
  schneier@counterpane.com (Bruce Schneier) wrote:

>
> >>[...]

> I'm about as tolerant as you can get, but the personal attacks on me
> have been going on for a lot longer than a day and are getting worse.
> I have just installed a newsreader with a killfile, though, and things
> are looking up.
>
> Bruce
>

How political correct. I guess that means he has not the balls
to follow through with the dough. So he can pretend not to read
all the posts. Bruce is a little man with a big EGO

He reads it!

--
http://cryptography.org/cgi-bin/crypto.cgi/Misc/scott19u.zip
ftp search for scott*u.zip in norway

----------== Posted via Deja News, The Discussion Network ==----------
http://www.dejanews.com/ Search, Read, Discuss, or Start Your Own

---

Subject: Re: Memo to the Amateur Cipher Designer
Date: 31 Oct 1998 00:36:10 GMT
From: jpeschel@aol.com (JPeschel)
Message-ID: <19981030193610.25670.00001657@ng34.aol.com>
References: <71dkmt$763$1@nnrp1.dejanews.com>
Newsgroups: sci.crypt
Lines: 17

dscott@networkusa.net writes:
> How political correct. I guess that means he has not the balls
>to follow through with the dough. So he can pretend not to read
>all the posts. Bruce is a little man with a big EGO

Look, Dave, you aren't helping your cause any by flinging personal
insults. Analyze his work -- Blowfish, TwoFish, etc., -- if you must
but we already got the idea you don't like Schneier.

Joe
_____

Joe Peschel
D.O.E. SysWorks
http://members.aol.com/jpeschel/index.htm
_____

Subject: Re: Memo to the Amateur Cipher Designer
Date: Sat, 31 Oct 1998 01:27:55 GMT
From: remo.a.linky@anagrams.r.us (Remo A. Linky)
Message-ID: <363a6605.428958859@news.alt.net>
References: <363a2037.500639@news.visi.com>
Newsgroups: sci.crypt
Lines: 21

schneier@counterpane.com (Bruce Schneier) wrote:

>I'm about as tolerant as you can get,

I noticed that.

>but the personal attacks on me
>have been going on for a lot longer than a day and are getting worse.

It's a foolish way to treat sci.crypt's most valuable contributor.

>I have just installed a newsreader with a killfile, though, and things
>are looking up.

You upgraded from Forte Free Agent to Forte Agent. Good move! It has a
powerful and versatile filter expression language that should make things
more livable for you here. Thanks for hanging in there!
--
"Remo A. Linky" better known as 2681.749530@mail.serve.com.
0123 4 56789 <- Use this key to decode my email address.
5 X 5 Poker - http://www.serve.com/games/

Subject: Re: Memo to the Amateur Cipher Designer
Date: Sat, 31 Oct 1998 17:23:36 GMT
From: schneier@counterpane.com (Bruce Schneier)
Message-ID: <363e477c.2403864@news.visi.com>
References: <363a6605.428958859@news.alt.net>
Newsgroups: sci.crypt
Lines: 30

On Sat, 31 Oct 1998 01:27:55 GMT, remo.a.linky@anagrams.r.us (Remo A.

Linky) wrote:

>schneier@counterpane.com (Bruce Schneier) wrote:
>
>>I'm about as tolerant as you can get,
>
>I noticed that.
>
>>but the personal attacks on me
>>have been going on for a lot longer than a day and are getting worse.
>
>It's a foolish way to treat sci.crypt's most valuable contributor.

Thanks.

>>I have just installed a newsreader with a killfile, though, and things
>>are looking up.
>
>You upgraded from Forte Free Agent to Forte Agent. Good move! It has a
>powerful and versatile filter expression language that should make things
>more livable for you here. Thanks for hanging in there!

It's working out already....

Bruce
****
Bruce Schneier, President, Counterpane Systems Phone: 612-823-1098
101 E Minnehaha Parkway, Minneapolis, MN 55419 Fax: 612-823-1590
Free crypto newsletter. See: http://www.counterpane.com

Subject: Re: Memo to the Amateur Cipher Designer
Date: Sat, 31 Oct 1998 03:59:13 GMT
From: dscott@networkusa.net
Message-ID: <71e1uh$nv8$1@nnrp1.dejanews.com>
References: <363a2037.500639@news.visi.com>
Newsgroups: sci.crypt
Lines: 29

In article <363a2037.500639@news.visi.com>,
  schneier@counterpane.com (Bruce Schneier) wrote:

 snip...

>
> I don't mean to insult you. I am just the bearer of news. You are
> free to be insulted. Maybe you are correct. I have found that it is
> better to play by the rules as they are than it is to complain about
> how unfair they are. I have found this to be true, even if the rules
> are unfair.
>
>
  snip....

  Lets exaime this GEM from the so called crypto god for what it
really means. It means I am recognized as on top. So even though
my rules are unfair to bad for you other guys and I don't mind
telling about these unfair rules of mine.

Of course I was parpharsing him by I (I meant Mr. B. S.)

--
http://cryptography.org/cgi-bin/crypto.cgi/Misc/scott19u.zip
ftp search for scott*u.zip in norway

----------== Posted via Deja News, The Discussion Network ==----------
http://www.dejanews.com/ Search, Read, Discuss, or Start Your Own

---

Subject: Re: Memo to the Amateur Cipher Designer
Date: Sat, 31 Oct 1998 02:14:18 -0600
From: jgfunj@EnqvbSerrGrknf.pbz (W T Shaw)
Message-ID: <jgfunj-3110980214190001@dialup168.itexas.net>
References: <363a2037.500639@news.visi.com>
Newsgroups: sci.crypt
Lines: 26

In article <363a2037.500639@news.visi.com>, schneier@counterpane.com
(Bruce Schneier) wrote:
>
> I'm about as tolerant as you can get, but the personal attacks on me
> have been going on for a lot longer than a day and are getting worse.
> I have just installed a newsreader with a killfile, though, and things
> are looking up.
>
That seems somehow consistent. But, please don't consider our academic
and philosophical criticisms as personal attacks; heated debate can merely
mean that people are getting down to the most important aspects of their
differences, those which they feel comfortable with and find painful to
analyze.

It is good that you have strong feelings about what you know; it could

mean that you are likely to be able to defend your position with
information you have considered already. However, if you have missed
something, it is best to hear from proponents of other points of view who
might be just as zealous for their own reasons. Truth is sometimes found
where no one previously stood, and all relish in the outcome.
--

Heard recently on Larry King: Jimmy Carter and Billy Graham
agreeing that it is sometimes wise to tell a lie.

Decrypt with ROT13 to get correct email address.

Subject: Re: Memo to the Amateur Cipher Designer
Date: Sat, 31 Oct 1998 17:25:23 GMT
From: schneier@counterpane.com (Bruce Schneier)
Message-ID: <363f47a9.2448333@news.visi.com>
References: <jgfunj-3110980214190001@dialup168.itexas.net>
Newsgroups: sci.crypt
Lines: 35

On Sat, 31 Oct 1998 02:14:18 -0600, jgfunj@EnqvbSerrGrknf.pbz (W T
Shaw) wrote:

>In article <363a2037.500639@news.visi.com>, schneier@counterpane.com
>(Bruce Schneier) wrote:
>>
>> I'm about as tolerant as you can get, but the personal attacks on me
>> have been going on for a lot longer than a day and are getting worse.
>> I have just installed a newsreader with a killfile, though, and things
>> are looking up.
>>
>That seems somehow consistent. But, please don't consider our academic
>and philosophical criticisms as personal attacks; heated debate can merely
>mean that people are getting down to the most important aspects of their
>differences, those which they feel comfortable with and find painful to
>analyze.

I'm not. I like the academic and philosophical criticisms; that's why
I stay here.

>It is good that you have strong feelings about what you know; it could
>mean that you are likely to be able to defend your position with
>information you have considered already. However, if you have missed
>something, it is best to hear from proponents of other points of view who
>might be just as zealous for their own reasons. Truth is sometimes found
>where no one previously stood, and all relish in the outcome.

Of course. It's the bad-English slightly-psychotic zero-content
personal abuse that I was tired of.

Bruce
****
Bruce Schneier, President, Counterpane Systems Phone: 612-823-1098
101 E Minnehaha Parkway, Minneapolis, MN 55419 Fax: 612-823-1590
Free crypto newsletter. See: http://www.counterpane.com

Subject: Re: Memo to the Amateur Cipher Designer
Date: Sat, 31 Oct 1998 02:01:34 -0600
From: jgfunj@EnqvbSerrGrknf.pbz (W T Shaw)
Message-ID: <jgfunj-3110980201340001@dialup168.itexas.net>
References: <363a1ada.12425654@news.io.com>
Newsgroups: sci.crypt
Lines: 18

In article <363a1ada.12425654@news.io.com>, ritter@io.com (Terry Ritter) wrote:
>
> If your paper is reviewed
> not by its content, but by where the original ideas came from, or what
> its references are, we are back to having a serious conflict with the
> meaning of Science.
>
Then, so much for inspiration. Answering the call for knowing exactly
where ideas might come from might cause some rather foolish sounding, be
they most important, references. Sometimes it is best to see how far you
can go on your own without tainting your thoughts with the misgivings of
others.
--

Heard recently on Larry King: Jimmy Carter and Billy Graham
agreeing that it is sometimes wise to tell a lie.

Decrypt with ROT13 to get correct email address.

Subject: Re: Memo to the Amateur Cipher Designer
Date: Fri, 30 Oct 1998 22:00:05 GMT
From: jsavard@tenMAPSONeerf.edmonton.ab.ca (John Savard)
Message-ID: <363a367c.19019214@news.prosurfr.com>
References: <3639d8bf.741333@news.visi.com>
Newsgroups: sci.crypt
Lines: 14

schneier@counterpane.com (Bruce Schneier) wrote, in part:

>Patents are not relevent to academic publication. There are academics
>who also patent. Ritter is an example of someone who does not publish
>(in an academic sense) but does patent. His writings are generally
>ignored by the academic community. I'm sorry this is true; I'd like
>it to be different.

It should be noted, though, that Terry Ritter has had academic
publications in the past; three or so papers in Cryptologia on Dynamic
Substitution and related topics.

John Savard
http://members.xoom.com/quadibloc/index.html

Subject: Re: Memo to the Amateur Cipher Designer
Date: Fri, 30 Oct 1998 23:06:10 GMT
From: schneier@counterpane.com (Bruce Schneier)
Message-ID: <363a4656.4232760@news.visi.com>
References: <363a367c.19019214@news.prosurfr.com>
Newsgroups: sci.crypt
Lines: 22

On Fri, 30 Oct 1998 22:00:05 GMT, jsavard@tenMAPSONeerf.edmonton.ab.ca
(John Savard) wrote:

>schneier@counterpane.com (Bruce Schneier) wrote, in part:
>
>>Patents are not relevent to academic publication. There are academics
>>who also patent. Ritter is an example of someone who does not publish
>>(in an academic sense) but does patent. His writings are generally
>>ignored by the academic community. I'm sorry this is true; I'd like
>>it to be different.
>
>It should be noted, though, that Terry Ritter has had academic
>publications in the past; three or so papers in Cryptologia on Dynamic
>Substitution and related topics.

Thanks for pointing that out. I had meant to, but forgot.

Bruce
****
Bruce Schneier, President, Counterpane Systems Phone: 612-823-1098
101 E Minnehaha Parkway, Minneapolis, MN 55419 Fax: 612-823-1590
Free crypto newsletter. See: http://www.counterpane.com

Subject: Re: Memo to the Amateur Cipher Designer
Date: Sat, 31 Oct 1998 03:51:34 GMT
From: dscott@networkusa.net
Message-ID: <71e1g6nae1@nnrp1.dejanews.com>
References: <363a4656.4232760@news.visi.com>
Newsgroups: sci.crypt
Lines: 33

In article <363a4656.4232760@news.visi.com>,
  schneier@counterpane.com (Bruce Schneier) wrote:
> On Fri, 30 Oct 1998 22:00:05 GMT, jsavard@tenMAPSONeerf.edmonton.ab.ca
> (John Savard) wrote:
>
> >schneier@counterpane.com (Bruce Schneier) wrote, in part:
> >
> >>Patents are not relevent to academic publication. There are academics
> >>who also patent. Ritter is an example of someone who does not publish
> >>(in an academic sense) but does patent. His writings are generally
> >>ignored by the academic community. I'm sorry this is true; I'd like
> >>it to be different.
> >
> >It should be noted, though, that Terry Ritter has had academic
> >publications in the past; three or so papers in Cryptologia on Dynamic
> >Substitution and related topics.
>
> Thanks for pointing that out. I had meant to, but forgot.
>

    Get real your iragance is showing through. NO WAY IN HELL
did you mean to. Just like clinton he wasn't sure he had sex.
You are a good politician. I doubt if you can hold a candle
to Ritter in the brains department though.

--
http://cryptography.org/cgi-bin/crypto.cgi/Misc/scott19u.zip
ftp search for scott*u.zip in norway

----------== Posted via Deja News, The Discussion Network ==----------
http://www.dejanews.com/ Search, Read, Discuss, or Start Your Own

---

Subject: Re: Memo to the Amateur Cipher Designer

Date: Sat, 31 Oct 1998 03:47:59 GMT
From: dscott@networkusa.net
Message-ID: <71e19g$n7t$1@nnrp1.dejanews.com>
References: <363a367c.19019214@news.prosurfr.com>
Newsgroups: sci.crypt
Lines: 38

In article <363a367c.19019214@news.prosurfr.com>,
  jsavard@tenMAPSONeerf.edmonton.ab.ca (John Savard) wrote:
> schneier@counterpane.com (Bruce Schneier) wrote, in part:
>
> >Patents are not relevent to academic publication. There are academics
> >who also patent. Ritter is an example of someone who does not publish
> >(in an academic sense) but does patent. His writings are generally
> >ignored by the academic community. I'm sorry this is true; I'd like
> >it to be different.
>
> It should be noted, though, that Terry Ritter has had academic
> publications in the past; three or so papers in Cryptologia on Dynamic
> Substitution and related topics.
>
> John Savard
> http://members.xoom.com/quadibloc/index.html
>

  John Bruce most likely was in his usual put people down mode
I guess he never reads things like Cryptologia. If Ritter gets
published in that it is a better honor than in some toy book
Brucey writes.

  Ritter is a person whose life is real crypto Bruce most
likely has people doing it for him so he can be a the front
man. I have never meet either man but it is obvious from
the posts of past Ritter is more up on what is happening
in real crypto. But Bruce is a better PR man. And when it
comes to money good PR wins hands down. But then again
this is just my humble opinion.

--
http://cryptography.org/cgi-bin/crypto.cgi/Misc/scott19u.zip
ftp search for scott*u.zip in norway

----------== Posted via Deja News, The Discussion Network ==----------
http://www.dejanews.com/ Search, Read, Discuss, or Start Your Own

---

Subject: Re: Memo to the Amateur Cipher Designer
Date: Sat, 31 Oct 1998 04:52:24 GMT
From: ritter@io.com (Terry Ritter)
Message-ID: <363a971b.2194234@news.io.com>
References: <71e19g$n7t$1@nnrp1.dejanews.com>
Newsgroups: sci.crypt
Lines: 19

On Sat, 31 Oct 1998 03:47:59 GMT, in
<71e19g$n7t$1@nnrp1.dejanews.com>, in sci.crypt dscott@networkusa.net
wrote . . . all sorts of stuff:

Time out guys! Try to relax. Different viewpoints can be irritating
but life would be pretty dull without them.

Or we can get mad and yell, but that won't change anything or anybody,
or drive anybody away, or create any groundswell of approval, so all
of that is just a waste of time for everybody.

Let's just try to hold it down a little.

Terry Ritter ritter@io.com http://www.io.com/~ritter/
Crypto Glossary http://www.io.com/~ritter/GLOSSARY.HTM

Subject: Re: Memo to the Amateur Cipher Designer
Date: Wed, 28 Oct 1998 22:00:27 GMT
From: ritter@io.com (Terry Ritter)
Message-ID: <363793f4.15728422@news.io.com>
References: <36373706.886670@news.visi.com>
Newsgroups: sci.crypt
Lines: 45

On Wed, 28 Oct 1998 15:32:00 GMT, in <36373706.886670@news.visi.com>,
in sci.crypt schneier@counterpane.com (Bruce Schneier) wrote:

>[...]
>There are Ritter's
>designs. Any of these algorithms could potentially be cryptanalyzed
>by amateurs.

In fact I *encourage* my designs to be broken.

The problem I pose in my designs is distinctly unlike the usual cryptanalysis situation: Normally, someone develops a large, complex, fixed design and says "break it." Such an attack generally takes a very significant investment of resources. Even then the best we can know is that the given design "was" or "was not" broken.

But my stuff is *scalable*: We have *no* *doubt* that the ultimate scaled-down versions will be weak. By using few component types and an overall regular structure, I hope to *expose* every avenue into the design. The very intent of this is to have some well-understood strength at tiny size. The hope of this strategy is the ability to extrapolate from one or more toy versions to real size strength values which have some science behind them. That would be a nice change.

Currently I display no cipher code, but of course the scaled-down ciphers will be special builds anyway. I obviously do have some internal prototypes (the ones aimed for AES before I was not allowed to participate) which I can test and measure. But technology development has continued and I expect that any new versions of these ciphers would use the keyed nonlinear mixings I described last month. In the Mixing designs, the only thing left which is not keyed is the FFT-style mixing pattern (and that could be).

A great deal can be learned by studying systems and attacks of the past. I give some references to stream cipher constructions and attacks in:

   http://www.io.com/~ritter/LEARNING.HTM#ForDesigners

Terry Ritter   ritter@io.com   http://www.io.com/~ritter/
Crypto Glossary   http://www.io.com/~ritter/GLOSSARY.HTM

Subject: Re: Memo to the Amateur Cipher Designer
Date: Wed, 28 Oct 1998 22:45:43 GMT
From: schneier@counterpane.com (Bruce Schneier)
Message-ID: <36379d96.2892303@news.visi.com>
References: <363793f4.15728422@news.io.com>
Newsgroups: sci.crypt
Lines: 70

On Wed, 28 Oct 1998 22:00:27 GMT, ritter@io.com (Terry Ritter) wrote:

>
>On Wed, 28 Oct 1998 15:32:00 GMT, in <36373706.886670@news.visi.com>,
>in sci.crypt schneier@counterpane.com (Bruce Schneier) wrote:
>
>>[...]
>>There are Ritter's
>>designs. Any of these algorithms could potentially be cryptanalyzed
>>by amateurs.
>
>In fact I *encourage* my designs to be broken.

Of couse. As any reasonable designer would.

>The problem I pose in my designs is distinctly unlike the usual
>cryptanalysis situation: Normally, someone develops a large, complex,
>fixed design and says "break it." Such an attack generally takes a
>very significant investment of resources. Even then the best we can
>know is that the given design "was" or "was not" broken.
>
>But my stuff is *scalable*: We have *no* *doubt* that the ultimate
>scaled-down versions will be weak. By using few component types and
>an overall regular structure, I hope to *expose* every avenue into the
>design. The very intent of this is to have some well-understood
>strength at tiny size. The hope of this strategy is the ability to
>extrapolate from one or more toy versions to real size strength values
>which have some science behind them. That would be a nice change.

But your designs are as analyzable as any others. It would be better if you would put up strawman designs, so that people would have something concrete to analyze: the idea being that the concrete analysis serves to illuminate something about the general design. But in some ways your stuff is easier for a beginner to look at; he can attack the toy versions successfully and learn from the results, and then hopefully take that knowledge to the more elaborate versions.

>Currently I display no cipher code, but of course the scaled-down
>ciphers will be special builds anyway. I obviously do have some
>internal prototypes (the ones aimed for AES before I was not allowed
>to participate)

(Don't take me there....)

>which I can test and measure. But technology
>development has continued and I expect that any new versions of these
>ciphers would use the keyed nonlinear mixings I described last month.
>In the Mixing designs, the only thing left which is not keyed is the
>FFT-style mixing pattern (and that could be).
>

>A great deal can be learned by studying systems and attacks of the
>past. I give some references to stream cipher constructions and
>attacks in:
>
> http://www.io.com/~ritter/LEARNING.HTM#ForDesigners
>
>Terry Ritter ritter@io.com http://www.io.com/~ritter/
>Crypto Glossary http://www.io.com/~ritter/GLOSSARY.HTM

By the way, thank you for not assuming that I was calling your designs
poor or weak or stupid or anything like that. I simply put your work
in the pile as something that has not received sufficient attention.
I didn't think you would misinterpret what I said, althoguh I got two
emails from people who warned me you would. Thank you again.

Bruce
****
Bruce Schneier, President, Counterpane Systems Phone: 612-823-1098
101 E Minnehaha Parkway, Minneapolis, MN 55419 Fax: 612-823-1590
Free crypto newsletter. See: http://www.counterpane.com

Subject: Re: Memo to the Amateur Cipher Designer
Date: Fri, 30 Oct 1998 22:06:31 GMT
From: jsavard@tenMAPSONeerf.edmonton.ab.ca (John Savard)
Message-ID: <363a374d.19228278@news.prosurfr.com>
References: <36379d96.2892303@news.visi.com>
Newsgroups: sci.crypt
Lines: 18

schneier@counterpane.com (Bruce Schneier) wrote, in part:

>I didn't think you would misinterpret what I said, althoguh I got two
>emails from people who warned me you would. Thank you again.

It was clear enough what you originally said, so I wouldn't think that
any reasonable person would misinterpret it: that there are these
various designs out there, and an amateur might consider looking at
them to try and cryptanalyze them.

But the quote as I saw it in later posts, out of context, seems to say
that here are designs that even amateurs can break. Thus, if Terry
Ritter, or someone else, had seen only the quote, and couldn't get at
the original post, (these things happen on USENET) misinterpretation
would have been excusable.

John Savard
http://members.xoom.com/quadibloc/index.html

Subject: Re: Memo to the Amateur Cipher Designer
Date: Wed, 28 Oct 1998 22:25:25 GMT
From: "Douglas A. Gwyn" <gwyn@arl.mil>
Message-ID: <363799D5.7DD4A347@arl.mil>
References: <3636F8F7.F8287D6B@stud.uni-muenchen.de>
Newsgroups: sci.crypt
Lines: 31

One point about the role of amateurs in cryptosystem design:
One can consider a body of knowledge (think: physics) as a domain for a
"goodness" function
that is more or less continuous with maxima at the "best" understanding.
As usual in optimization problems, one can search for the maximum via
"hill-climbing" methods,
but there is the danger of converging to a local maximum rather than the
higher global maximum.
Thus, a continual evolution of knowledge by incremental improvements
might reach stability
but completely fail to attain a better understanding; in applications of
hill-climbing,
one sometimes resorts to a "scattering" of starting points searched in
parallel,
or an occasional step in what temporarily appears to be a detour, with
the hope that
the problem of convergence to local maxima is thereby reduced and thus
that better results are found.
(One improves the odds of attaining the true global maximum.)
Similarly, occasionally breakthroughs in knowledge are made by
"dabblers" who work outside the
mainstream. However, the odds are still that in any given case a
dabbler is wasting his time.
It's just good for the overall evolution of science that there are a lot
of dabblers.
Unfortunately, in fields like particle physics, dabblers aren't even
able to get on the playing field;
thus those fields tend to get into a rut where everybody sings from the
same sheet of music
and never realize that it's the wrong song, because there is no
corrective factor.

Subject: Re: Memo to the Amateur Cipher Designer
Date: Thu, 29 Oct 1998 15:55:28 -0600

From: jgfunj@EnqvbSerrGrknf.pbz (W T Shaw)
Message-ID: <jgfunj-2910981555280001@dialup105.itexas.net>
References: <36386EBF.E621AB07@egg.chips.and.spam.com>
<19981027172831.09766.00000187@ng26.aol.com>
<36363316.8304318@news.visi.com>
Newsgroups: sci.crypt
Lines: 35

In article <36386EBF.E621AB07@egg.chips.and.spam.com>, fungus
<spam@egg.chips.and.spam.com> wrote:

> JPeschel wrote:
> >
> > schneier@counterpane.com writes:
> > >Is there an interest in
> > >finding an algorithm and, as a group, cryptanalyzing it?
> >
> > I can think of one algorithm that might interest the group:
> > scott16!
> >
>
> No, please...!
>
> If we break scott16 or scott19 then all that will happen
> is that a scott21 will appear, and we'll be back to square
> one....
>
> Don't believe me? Ask what happened to scott14...
>
Don't you believe in evolution? Advances are allowed to be vertical as
well as horizontal, which means that one can improve on his own designs
and others can cherry pick and cull out the obvious bad aspects. However,
I'm not sure what the dressed weight of one of his ciphers would be after
all the waste was trimmed; there may actually be something worthwhile,
worth crediting him, maybe not; full analysis might tell.

I suppose I am an optomist, but fairness seems to mean giving someone the
benefit of the doubt, whether you want to or not.
--

Please excuse if there are multiple postings for my responses...I have no idea where
they come from as I only receive one confirmation for each posting from my
newsserver.

Decrypt with ROT13 to get correct email address.

Subject: Re: Memo to the Amateur Cipher Designer
Date: Thu, 29 Oct 1998 20:59:10 GMT
From: schneier@counterpane.com (Bruce Schneier)
Message-ID: <3638d6d7.22562148@news.visi.com>
References: <jgfunj-2910981555280001@dialup105.itexas.net>
Newsgroups: sci.crypt
Lines: 26

On Thu, 29 Oct 1998 15:55:28 -0600, jgfunj@EnqvbSerrGrknf.pbz (W T
Shaw) wrote:
>> If we break scott16 or scott19 then all that will happen
>> is that a scott21 will appear, and we'll be back to square
>> one....
>>
>> Don't believe me? Ask what happened to scott14...
>>
>Don't you believe in evolution? Advances are allowed to be vertical as
>well as horizontal, which means that one can improve on his own designs
>and others can cherry pick and cull out the obvious bad aspects. However,
>I'm not sure what the dressed weight of one of his ciphers would be after
>all the waste was trimmed; there may actually be something worthwhile,
>worth crediting him, maybe not; full analysis might tell.
>
>I suppose I am an optomist, but fairness seems to mean giving someone the
>benefit of the doubt, whether you want to or not.

If someone tries hard enough, he will wear his "benefit of the doubt"
out. I suspect that is what is happening in this case.

Bruce
****
Bruce Schneier, President, Counterpane Systems Phone: 612-823-1098
101 E Minnehaha Parkway, Minneapolis, MN 55419 Fax: 612-823-1590
Free crypto newsletter. See: http://www.counterpane.com

Subject: Re: Memo to the Amateur Cipher Designer
Date: Fri, 30 Oct 1998 01:49:25 GMT
From: dscott@networkusa.net
Message-ID: <71b5v5$s49$1@nnrp1.dejanews.com>
References: <3638d6d7.22562148@news.visi.com>
Newsgroups: sci.crypt
Lines: 51

In article <3638d6d7.22562148@news.visi.com>,
  schneier@counterpane.com (Bruce Schneier) wrote:
> On Thu, 29 Oct 1998 15:55:28 -0600, jgfunj@EnqvbSerrGrknf.pbz (W T

> Shaw) wrote:
> >> If we break scott16 or scott19 then all that will happen
> >> is that a scott21 will appear, and we'll be back to square
> >> one....
> >>
> >> Don't believe me? Ask what happened to scott14...
> >>
> >Don't you believe in evolution? Advances are allowed to be vertical as
> >well as horizontal, which means that one can improve on his own designs
> >and others can cherry pick and cull out the obvious bad aspects. However,
> >I'm not sure what the dressed weight of one of his ciphers would be after
> >all the waste was trimmed; there may actually be something worthwhile,
> >worth crediting him, maybe not; full analysis might tell.
> >
> >I suppose I am an optomist, but fairness seems to mean giving someone the
> >benefit of the doubt, whether you want to or not.
>
> If someone tries hard enough, he will wear his "benefit of the doubt"
> out. I suspect that is what is happening in this case.
>

OK Bruce I hope others read your response to re:book recom
since it was the only time I think you may have thrown a bone
my way. The question is was it full of meat or so empty as to
be marrowless?

Not sure if I should give you the benefit of the doubt
since you only tend to look down on the masses and prevent
any one with a good idea to get a fair chance. Look
at my respose to your response to me the ball is in your
court.

> Bruce
> ***
> Bruce Schneier, President, Counterpane Systems Phone: 612-823-1098
> 101 E Minnehaha Parkway, Minneapolis, MN 55419 Fax: 612-823-1590
> Free crypto newsletter. See: http://www.counterpane.com
>

--
http://cryptography.org/cgi-bin/crypto.cgi/Misc/scott19u.zip
ftp search for scott*u.zip in norway

----------== Posted via Deja News, The Discussion Network ==----------
http://www.dejanews.com/ Search, Read, Discuss, or Start Your Own

---

Subject: Re: Memo to the Amateur Cipher Designer
Date: Sat, 31 Oct 1998 07:53:18 +0100
From: fungus <spam@egg.chips.and.spam.com>
Message-ID: <363AB3DE.B3C3378D@egg.chips.and.spam.com>
References: <jgfunj-2910981555280001@dialup105.itexas.net>
Newsgroups: sci.crypt
Lines: 26

W T Shaw wrote:
>
> > Don't believe me? Ask what happened to scott14...
> >
>
> I suppose I am an optomist, but fairness seems to mean giving
> someone the benefit of the doubt, whether you want to or not.
>

A serious analysis of something like ScottXX is a serious
undertaking, probably a couple of months of hard work.

Given that we can be fairly sure about the outcome (based
on personal experience with Mr Scott) nobody's volunteering
to do this.

Are you?

Read a few of Mr Scott's postings before answering...

--
<___/>
/ O O \
_____/ FTB.

Subject: Re: Memo to the Amateur Cipher Designer
Date: Mon, 09 Nov 1998 04:45:45 -1000
From: newWebsite <ww@wW.com>
Message-ID: <36470019.53DE@wW.com>
References: <N910589592.4879@ruby.ansuz.sooke.bc.ca>
  <363AB3DE.B3C3378D@egg.chips.and.spam.com>
Newsgroups: sci.crypt

Lines: 15

```
> For whatever it's worth: whenever the web site with the readable algorithm
> descriptions is up, I plan to make a serious examination of the scottNu
> algorithms.  I am not a professional nor even a serious amateur
> cryptanalyst, just a programmer who likes encryption, and I don't really
> expect to be able to crack the system.  That's a comment on my own
> abilities, not on the strength of the system!  But I think it'd be
> interesting to look at, and should I find a weakness in it, I think the
> bragging rights and feeling of achievement would be adequate reward; I
> have no expectation of getting any money from the cipher designer.
> --
> The third girl had an upside-down penguin on       Matthew Skala
```

The website is ready for you!

http://members.xoom.com/ecil/

Terry Ritter, his *current address*, and his *top page*.

Last updated: 1999-01-20

The Meaning of "Break"

A *Ciphers By Ritter* Page

A discussion of what it means to claim a cipher is "broken."

Contents

- 1998-11-6 Hiroshi Fujita: "There are many different terms for attacks (ciphertext-only and so on), but only one term for success (break)."
- 1998-11-06 STL137: "'Theoretical' is a personal opinion...."
- 1998-11-06 Karl-Friedrich Lenz: "...in quite a lot of cases the distinction will be not only a matter of opinion."
- 1998-11-06 Andreas Enterrottacher: "I think your 'theoretical break' is an successful attack...but not a break.... Your 'practical break' would simply be called 'break'."
- 1998-11-06 Karl-Friedrich Lenz: "Look at Ritter's glossary: He defines break as the result of a successful attack."
- 1998-11-06 Andreas Enterrottacher: "I think this definition is not the commonly used one...."
- 1998-11-10 Terry Ritter: "The issue seems to be a conflict between the normal English understanding of 'broken' as 'impractical to use,' versus the crypto, where 'broken' does not necessarily imply that a cipher is impractical."
- 1998-11-10 Andreas Enterrottacher: "Every attack faster than brute force is successful. It may or may not be a break...."
- 1998-11-08 Bruce Schneier: "There is much abuse of the term 'break' in the literature." "Why do we care about impractical breaks? Because we are constantly required to make decisions about which ciphers to use."
- 1998-11-10 Terry Ritter: "I claim that cryptanalysis provides no information at all beyond a break or argument of weakness. A cipher without an academic break is not necessarily stronger, it just has yet to be broken."
- 1998-11-10 Frank Gifford: "Wouldn't you then agree that avoiding a cipher because of an unknown future attack is also irrational?"
- 1998-11-10 Terry Ritter: "We do avoid ciphers which seem like they *might* have some particular attack strategy.... But without such an argument and without practical weakness, avoiding the cipher BECAUSE IT IS WEAK would seem irrational, yes."
- 1998-11-12 dianelos@tecapro.com: "Several of the AES candidates include features designed specifically as a possible defense against unknown attacks. Unknown future attacks are a real threat and should be a present worry."
- 1998-11-10 Frank Gifford: "...it's possible that what is a theoretical attack today could be tweaked by others in the crypto community and make it into a viable attack for tomorrow."
- 1998-11-10 Bruce Schneier: "'Attacks always get better.'"
- 1998-11-10 Terry Ritter: "a theoretical attack could be seen as a clear bill of health for a cipher with respect to that attack, unless there is some particular basis for an extrapolation of weakness."
- 1998-11-10 Bruce Schneier: "I just don't agree. And I am willing to disagree."

```
Subject: Many attacks, one break?
Date: Fri, 6 Nov 1998 15:42:01 +0900
From: "Hiroshi Fujita" &LT;lenz@als.aoyama.ac.jp>
Message-ID: &LT;71u5q4$fcf$1@ac-nws.cc.aoyama.ac.jp>
Newsgroups: sci.crypt
Lines: 17

There are many different terms for attacks (ciphertext-only and so on), but
only one term for success (break).

Why not use the same convention as for attacks when talking about success?
That is, ciphertext-only break, known-plaintext break and so on.

I would also like to propose the terms theoretical break (like reducing a
workload from 128 bits bruteforce to 110 bits bruteforce) and practical
break (obtaining any information on key or message in real life).

Karl-Friedrich Lenz :-)
www.toptext.com
```

```
Subject: Re: Many attacks, one break?
Date: 6 Nov 1998 07:30:16 GMT
From: stl137@aol.com (STL137)
Message-ID: &LT;19981106023016.29961.00001044@ng34.aol.com>
References: &LT;71u5q4$fcf$1@ac-nws.cc.aoyama.ac.jp>
Newsgroups: sci.crypt
Lines: 9

"Theoretical" is a personal opinion, not really quantifiable. (If you put the
limit somewhere, I can either A. Show you how more funding can break it, or B.
Show you how no one could have that kind of funding".
------
STL137@aol.com  ===>  Website: http://members.aol.com/stl137/
PGP keys: ~~~pgp.html Quotes: ~~~quotes.html
"I have sworn upon the altar of God eternal hostility against every form of
tyranny over the mind of man" - Thomas Jefferson
```

```
Subject: Re: Many attacks, one break?
Date: 6 Nov 1998 03:45:08 -0800
From: Karl-Friedrich Lenz &LT;Karl-Friedrich@newsguy.com>
Message-ID: &LT;71ung4$mb5@edrn.newsguy.com>
References: &LT;19981106023016.29961.00001044@ng34.aol.com>
Newsgroups: sci.crypt
Lines: 17

In article , stl137@aol.com says...
>
>"Theoretical" is a personal opinion, not really quantifiable. (If you put the
>limit somewhere, I can either A. Show you how more funding can break it, or B.
>Show you how no one could have that kind of funding".

If you can show that no one could have the funding to brute force 110 bits, the
break of reducing 128 to 110 (which is considered a break by Bruce Schneier in
his cryptanalysis self-study course) would clearly be a theoretical break. On
the other hand, reading plaintext in a ciphertext-only attack against some
simple historical system or a system some newbie amateur came up with would
clearly be a practical break.
So while there might be borderline cases, in quite a lot of cases the
distinction will be not only a matter of opinion.

Karl-Friedrich Lenz
www.toptext.com/crypto/
```

```
Subject: Re: Many attacks, one break?
Date: Fri, 6 Nov 1998 09:32:55 +0100
From: &LT;tbb03ar@mail.lrz-muenchen.de>
Message-ID: &LT;Pine.GSO.4.03.9811060816420.513-100000@sun5.lrz-muenchen.de>
References: &LT;71u5q4$fcf$1@ac-nws.cc.aoyama.ac.jp>
Newsgroups: sci.crypt
Lines: 37

On Fri, 6 Nov 1998, Hiroshi Fujita wrote:

> There are many different terms for attacks (ciphertext-only and so on), but
> only one term for success (break).
>
> Why not use the same convention as for attacks when talking about success?
> That is, ciphertext-only break, known-plaintext break and so on.
>
> I would also like to propose the terms theoretical break (like reducing a
> workload from 128 bits bruteforce to 110 bits bruteforce) and practical
> break (obtaining any information on key or message in real life).

I think your 'theoretical break' is an successful attack (like the
known-plaintext-attacks against DES) but not a break (DES was never
broken). Your 'practical break' would simply be called 'break'.
Such a break may need some known or even chosen plaintext
(every algorithm should be strong against attacks with some kilobytes of
known plaintext, stronger ones should be strong against attacks with
megabytes or even gigabytes of chosen plaintext).


Andreas Enterrottacher

enterrottacher@lrz.tu-muenchen.de
enterrottacher@t-online.de

>
> Karl-Friedrich Lenz :-)
> www.toptext.com
>
>
>
>
>
>
>
```

```
Subject: Re: Many attacks, one break?
Date: 6 Nov 1998 03:38:31 -0800
From: Karl-Friedrich Lenz &LT;Karl-Friedrich@newsguy.com>
Message-ID: &LT;71un3n$lvv@edrn.newsguy.com>
References: &LT;Pine.GSO.4.03.9811060816420.513-100000@sun5.lrz-muenchen.de>
Newsgroups: sci.crypt
Lines: 16

In article ,
>I think your 'theoretical break' is an successful attack (like the
>known-plaintext-attacks against DES) but not a break (DES was never
>broken). Your 'practical break' would simply be called 'break'.

Look at Ritter's glossary: He defines break as the result of a successful
attack. In that definition, there is no difference between successful attack and
break, contrary to your usage.

But leaving words aside, we seem to agree that it does make a difference if a
break has real-world applications or is only of theoretical value. Many so
called breaks of modern ciphers seem to be quite far away from really reading
ciphertext or getting keys from a reasonable amount of known plaintext.

Karl-Friedrich Lenz
www.toptext.com/crypto/
```

```
Subject: Re: Many attacks, one break?
Date: Fri, 6 Nov 1998 14:32:04 +0100
From: &LT;tbb03ar@mail.lrz-muenchen.de>
Message-ID: &LT;Pine.GSO.4.03.9811061428550.10463-100000@sun5.lrz-muenchen.de>
References: &LT;71un3n$lvv@edrn.newsguy.com>
Newsgroups: sci.crypt
Lines: 35

On 6 Nov 1998, Karl-Friedrich Lenz wrote:

> In article ,
> >I think your 'theoretical break' is an successful attack (like the
> >known-plaintext-attacks against DES) but not a break (DES was never
> >broken). Your 'practical break' would simply be called 'break'.
>
> Look at Ritter's glossary: He defines break as the result of a successful
> attack. In that definition, there is no difference between successful attack and
> break, contrary to your usage.

I think this definition is not the commonly used one: This would imply
that as well DES (because of Bihams attack) as IDEA (don't know who was
the attacker) were broken, but practically both algorithms are secure
(except the small keyspace of DES).

>
> But leaving words aside, we seem to agree that it does make a difference if a
> break has real-world applications or is only of theoretical value. Many so
> called breaks of modern ciphers seem to be quite far away from really reading
> ciphertext or getting keys from a reasonable amount of known plaintext.

Accepted.

>
> Karl-Friedrich Lenz
> www.toptext.com/crypto/
>
>

Andreas Enterrottacher

enterrottacher@lrz.tu-muenchen.de
enterrottacher@t-online.de
```

```
Subject: Re: Many attacks, one break?
Date: Tue, 10 Nov 1998 03:34:42 GMT
From: ritter@io.com (Terry Ritter)
Message-ID: &LT;3647b447.4474700@news.io.com>
References: &LT;Pine.GSO.4.03.9811061428550.10463-100000@sun5.lrz-muenchen.de>
Newsgroups: sci.crypt
Lines: 73


On Fri, 6 Nov 1998 14:32:04 +0100, in
&LT;Pine.GSO.4.03.9811061428550.10463-100000@sun5.lrz-muenchen.de>, in
sci.crypt &LT;tbb03ar@mail.lrz-muenchen.de> wrote:

>On 6 Nov 1998, Karl-Friedrich Lenz wrote:
>
>> In article ,
>> >I think your 'theoretical break' is an successful attack (like the
>> >known-plaintext-attacks against DES) but not a break (DES was never
>> >broken). Your 'practical break' would simply be called 'break'.
>>
>> Look at Ritter's glossary: He defines break as the result of a successful
>> attack. In that definition, there is no difference between successful attack and
>> break, contrary to your usage.
>
>I think this definition is not the commonly used one: This would imply
>that as well DES (because of Bihams attack) as IDEA (don't know who was
>the attacker) were broken, but practically both algorithms are secure
>(except the small keyspace of DES).

From the Handbook of Applied Cryptography:

"1.23 Definition.  An encryption scheme is said to be *breakable* if a
third party, without prior knowledge of the key pair (e,d), can
systematically recover plaintext from corresponding ciphertext in some
appropriate time frame." [p.14]

...and...

"*Breaking* an information security service (which often involves more
than simply encryption) implies defeating the objective of the
intended service." [p.15]


1. It seems to me that "break" needs to apply to an arbitrary attack,
including attacks on hashing.  Thus: "a successful attack."

2. The issue seems to be a conflict between the normal English
understanding of "broken" as "impractical to use," versus the crypto,
where "broken" does not necessarily imply that a cipher is
impractical.  This is a real confusion and a slur on a good cipher.
But it would be very hard to state at just what strength a cipher *is*
no longer "useful":  Even weak ciphers can be effective and *useful*
in low-bandwidth situations.

One alternative would seem to be to reserve "break" for "a practical
attack," but then we have to define "practical."  Or a "break" could
be "the current best attack," but that would mean that past breaks
would no longer be breaks.  Or we could require that a "break" be
better than the advertised keyspace, but that still means we can have
very powerful "broken" ciphers and so does not solve the problem.

Presumably we can clarify the situation by using "practical break" to
express the sentiment of a serious weakness.  In the opposite
direction we might have: "impractical break," "academic break,"
"theoretical break," or "certificational break."  So the designer of a
slurred cipher could retort: "that 'break' was only academic; the
cipher is still secure in practice."

On the other hand, a "break" that doesn't work is no break at all.

3. One thing I *have* thought to do is to require that a "break" be
less effort than the advertised strength.  But then what do we call a
successful attack that takes *more* effort?  Is that a "failed"
attack?  Obviously not.  It is just a (currently) useless break.  So I
think we stay where we are.

---
Terry Ritter   ritter@io.com   http://www.io.com/~ritter/
Crypto Glossary   http://www.io.com/~ritter/GLOSSARY.HTM
```

```
Subject: Re: Many attacks, one break?
Date: Tue, 10 Nov 1998 11:43:43 +0100
From: &LT;tbb03ar@mail.lrz-muenchen.de>
Message-ID: &LT;Pine.GSO.4.03.9811100818350.11312-100000@sun5.lrz-muenchen.de>
References: &LT;3647b447.4474700@news.io.com>
Newsgroups: sci.crypt
Lines: 110

On Tue, 10 Nov 1998, Terry Ritter wrote:

>
> On Fri, 6 Nov 1998 14:32:04 +0100, in
> &LT;Pine.GSO.4.03.9811061428550.10463-100000@sun5.lrz-muenchen.de>, in
> sci.crypt &LT;tbb03ar@mail.lrz-muenchen.de> wrote:
>
> >On 6 Nov 1998, Karl-Friedrich Lenz wrote:
> >
> >> In article ,
> >> >I think your 'theoretical break' is an successful attack (like the
> >> >known-plaintext-attacks against DES) but not a break (DES was never
> >> >broken). Your 'practical break' would simply be called 'break'.
> >>
> >> Look at Ritter's glossary: He defines break as the result of a successful
> >> attack. In that definition, there is no difference between successful attack and
> >> break, contrary to your usage.
> >
> >I think this definition is not the commonly used one: This would imply
> >that as well DES (because of Bihams attack) as IDEA (don't know who was
> >the attacker) were broken, but practically both algorithms are secure
> >(except the small keyspace of DES).
>
> From the Handbook of Applied Cryptography:
>
> "1.23 Definition.  An encryption scheme is said to be *breakable* if a
> third party, without prior knowledge of the key pair (e,d), can
> systematically recover plaintext from corresponding ciphertext in some
> appropriate time frame." [p.14]
>
> ...and...
>
> "*Breaking* an information security service (which often involves more
> than simply encryption) implies defeating the objective of the
> intended service." [p.15]
>
>

The first definition says clearly 'in some appropriate time frame'.
This frame may be different for different ciphers - days for some ones,
decades for others, but this is clearly a 'practical break'.

The other one tells about the same with 'defeating the objective of the
intended service': An attack may allow to reduce the effective keysize
but not to break the cipher - who cares if the keysize of GOST could be
reduced to 192 bit?

> 1. It seems to me that "break" needs to apply to an arbitrary attack,
> including attacks on hashing.  Thus: "a successful attack."

Accepted.

>
> 2. The issue seems to be a conflict between the normal English
> understanding of "broken" as "impractical to use," versus the crypto,
> where "broken" does not necessarily imply that a cipher is
> impractical.  This is a real confusion and a slur on a good cipher.
> But it would be very hard to state at just what strength a cipher *is*
> no longer "useful":  Even weak ciphers can be effective and *useful*
> in low-bandwidth situations.
>
> One alternative would seem to be to reserve "break" for "a practical
> attack,"

Exactly this is done in the definitions you mentioned :-)

> but then we have to define "practical."

And this depends on the usage of the cipher.
I think, all ciphers that couldn't be broken with the present technology
within - let's say a century - is strong (first definition).
To call a cipher 'broken' it would (in addition) be neccessary to break it
faster than with brute force (meets as well first as second defintion).

RC4 with 40 bits keysize is weak but not broken.
DES is weak but not broken.


> ...
> 3. One thing I *have* thought to do is to require that a "break" be
> less effort than the advertised strength.  But then what do we call a
> successful attack that takes *more* effort?  Is that a "failed"
> attack?  Obviously not.  It is just a (currently) useless break.  So I
> think we stay where we are.

Every attack faster than brute force is successful.
It may or may not be a break - depending on the neccessary time/computer
power needed for a break.

For weak ciphers I think every successful attack is a break while for a
stronger one it is only a break if the speed of the attack allows to break
the cipher within acceptable time - let's say a century.

I don't see why anybody should use weak ciphers (except existing laws) in
a time where ciphers exist that are at the same time strong and fast
(at least I don't know of any successful attack on RC4, for example).

I'd prefer usage of key-escrow instead of using 40 bit ciphers (why use
something that could be broken by everybody instead of using something
that can be broken only by the government) while strong ciphers that
aren't escrowed are of course the better choice.

> ...


Andreas Enterrottacher

enterrottacher@lrz.tu-muenchen.de
enterrottacher@t-online.de
```

```
Subject: Re: Many attacks, one break?
Date: Sun, 08 Nov 1998 22:09:22 GMT
From: schneier@counterpane.com (Bruce Schneier)
Message-ID: &LT;36461506.7272610@news.visi.com>
References: &LT;71u5q4$fcf$1@ac-nws.cc.aoyama.ac.jp>
Newsgroups: sci.crypt
Lines: 48

On Fri, 6 Nov 1998 15:42:01 +0900, "Hiroshi Fujita"
&LT;lenz@als.aoyama.ac.jp> wrote:

>There are many different terms for attacks (ciphertext-only and so on), but
>only one term for success (break).
>
>Why not use the same convention as for attacks when talking about success?
>That is, ciphertext-only break, known-plaintext break and so on.
>
>I would also like to propose the terms theoretical break (like reducing a
>workload from 128 bits bruteforce to 110 bits bruteforce) and practical
>break (obtaining any information on key or message in real life).

You have a real complaint.  There is much abuse of the term "break" in
the literature.  For some, a "break" means that you can read traffic
encrypted in the cipher in an operational setting.  For others, a
"break" is demonstrating that one of more of the security claims of
the cipher is false.  I have heard the terms "theoretical break" or
"academic break" to describe the latter.

For example, an attack against a 128-bit key that has a workfactor of
2^110 is clearly impractical, but does mean that the 128-bit cipher is
providing less security than advertised.  This is an academic break.

There are also attacks against simplified versions of ciphers.  For
example, the current best attack against IDEA works against 4.5 rounds
(the complete cipher is 8 rounds).  The attack itself is impractical,
but even if it weren't, it wouldn't be able to decrypt any traffic
encrypted with the full 8 rounds of IDEA.

Why do we care about impractical breaks?  Because we are constantly
required to make decisions about which ciphers to use.  Given that we
don't know how to formulate proofs of security for block ciphers, all
we can do is pick the best we can of the available pile.  And that
means disgarding ciphers that have academic breaks if there are other,
also well studied, ciphers that don't.

This will become important in the AES selection process.  There are
fifteen candidates.  If an attack is discovered against a candidate,
even if it is a theoretical attack, it will likely knock that cipher
out of the running as long as there are alternatives that prevent such
attacks.

Bruce
**********************************************************************
Bruce Schneier, President, Counterpane Systems     Phone: 612-823-1098
101 E Minnehaha Parkway, Minneapolis, MN  55419      Fax: 612-823-1590
           Free crypto newsletter.  See:  http://www.counterpane.com
```

```
Subject: Re: Many attacks, one break?
Date: Tue, 10 Nov 1998 04:38:12 GMT
From: ritter@io.com (Terry Ritter)
Message-ID: &LT;3647c322.8278074@news.io.com>
References: &LT;36461506.7272610@news.visi.com>
Newsgroups: sci.crypt
Lines: 34


On Sun, 08 Nov 1998 22:09:22 GMT, in &LT;36461506.7272610@news.visi.com>,
in sci.crypt schneier@counterpane.com (Bruce Schneier) wrote:

>[...]
>Why do we care about impractical breaks?  Because we are constantly
>required to make decisions about which ciphers to use.  Given that we
>don't know how to formulate proofs of security for block ciphers, all
>we can do is pick the best we can of the available pile.  And that
>means disgarding ciphers that have academic breaks if there are other,
>also well studied, ciphers that don't.

I claim that cryptanalysis provides no information at all beyond a
break or argument of weakness.

A cipher without an academic break is not necessarily stronger, it
just has yet to be broken.  One could even argue that the broken
cipher is a better bet because it is better understood.


>This will become important in the AES selection process.  There are
>fifteen candidates.  If an attack is discovered against a candidate,
>even if it is a theoretical attack, it will likely knock that cipher
>out of the running as long as there are alternatives that prevent such
>attacks.

Without a deeper argument of weakness, avoiding a cipher because of an
impractical attack would seem irrational.

---
Terry Ritter   ritter@io.com   http://www.io.com/~ritter/
Crypto Glossary   http://www.io.com/~ritter/GLOSSARY.HTM
```

```
Subject: Re: Many attacks, one break?
Date: 10 Nov 1998 11:02:54 -0500
From: giff@eng.us.uu.net (Frank Gifford)
Message-ID: &LT;729o3e$eqm@perrier.eng.us.uu.net>
References: &LT;3647c322.8278074@news.io.com>
Newsgroups: sci.crypt
Lines: 21

In article &LT;3647c322.8278074@news.io.com>, Terry Ritter &LT;ritter@io.com> wrote:
>
>On Sun, 08 Nov 1998 22:09:22 GMT, in &LT;36461506.7272610@news.visi.com>,
>in sci.crypt schneier@counterpane.com (Bruce Schneier) wrote:
>
>>This will become important in the AES selection process.  There are
>>fifteen candidates.  If an attack is discovered against a candidate,
>>even if it is a theoretical attack, it will likely knock that cipher
>>out of the running as long as there are alternatives that prevent such
>>attacks.
>
>Without a deeper argument of weakness, avoiding a cipher because of an
>impractical attack would seem irrational.

Wouldn't you then agree that avoiding a cipher because of an unknown future
attack is also irrational?

-Giff

--
giff@uu.net       Too busy for a .sig
```

```
Subject: Re: Many attacks, one break?
Date: Tue, 10 Nov 1998 18:43:44 GMT
From: ritter@io.com (Terry Ritter)
Message-ID: &LT;36488954.5026441@news.io.com>
References: &LT;729o3e$eqm@perrier.eng.us.uu.net>
Newsgroups: sci.crypt
Lines: 26


On 10 Nov 1998 11:02:54 -0500, in &LT;729o3e$eqm@perrier.eng.us.uu.net>,
in sci.crypt giff@eng.us.uu.net (Frank Gifford) wrote:

>In article &LT;3647c322.8278074@news.io.com>, Terry Ritter &LT;ritter@io.com> wrote:
>>[...]
>>Without a deeper argument of weakness, avoiding a cipher because of an
>>impractical attack would seem irrational.
>
>Wouldn't you then agree that avoiding a cipher because of an unknown future
>attack is also irrational?

I don't see your point.

We do avoid ciphers which seem like they *might* have some particular
attack strategy -- I suppose that would be an extrapolation argument.


But without such an argument and without practical weakness, avoiding
the cipher BECAUSE IT IS WEAK would seem irrational, yes.

---
Terry Ritter   ritter@io.com   http://www.io.com/~ritter/
Crypto Glossary   http://www.io.com/~ritter/GLOSSARY.HTM
```

```
Subject: Re: Many attacks, one break?
Date: Thu, 12 Nov 1998 05:11:05 GMT
From: dianelos@tecapro.com
Message-ID: &LT;72dql9$2nr$1@nnrp1.dejanews.com>
References: &LT;729o3e$eqm@perrier.eng.us.uu.net>
Newsgroups: sci.crypt
Lines: 24

In article &LT;729o3e$eqm@perrier.eng.us.uu.net>,
  giff@eng.us.uu.net (Frank Gifford) wrote:
> In article &LT;3647c322.8278074@news.io.com>, Terry Ritter &LT;ritter@io.com>
wrote:
> >...
> >Without a deeper argument of weakness, avoiding a cipher because of an
> >impractical attack would seem irrational.
>
> Wouldn't you then agree that avoiding a cipher because of an unknown future
> attack is also irrational?

    Several of the AES candidates include features designed
    specifically as a possible defense against unknown attacks.
    Unknown future attacks are a real threat and should be a present
    worry. If all other factors are comparable, I think it does make
    sense to avoid a cipher where unknown attacks have played no role
    in its design.


--
http://www.tecapro.com
email: dianelos@tecapro.com

-----------== Posted via Deja News, The Discussion Network ==----------
http://www.dejanews.com/       Search, Read, Discuss, or Start Your Own
```

```
Subject: Re: Many attacks, one break?
Date: 10 Nov 1998 11:07:39 -0500
From: giff@eng.us.uu.net (Frank Gifford)
Message-ID: &LT;729ocb$esk@perrier.eng.us.uu.net>
References: &LT;3647c322.8278074@news.io.com>
Newsgroups: sci.crypt
Lines: 17

>On Sun, 08 Nov 1998 22:09:22 GMT, in &LT;36461506.7272610@news.visi.com>,
>in sci.crypt schneier@counterpane.com (Bruce Schneier) wrote:
>
>>This will become important in the AES selection process.  There are
>>fifteen candidates.  If an attack is discovered against a candidate,
>>even if it is a theoretical attack, it will likely knock that cipher
>>out of the running as long as there are alternatives that prevent such
>>attacks.

More to the point, it's possible that what is a theoretical attack today
could be tweaked by others in the crypto community and make it into a
viable attack for tomorrow.

-Giff

--
giff@uu.net       Too busy for a .sig
```

```
Subject: Re: Many attacks, one break?
Date: Tue, 10 Nov 1998 18:29:32 GMT
From: schneier@counterpane.com (Bruce Schneier)
Message-ID: &LT;364b85ff.2098415@news.visi.com>
References: &LT;729ocb$esk@perrier.eng.us.uu.net>
Newsgroups: sci.crypt
Lines: 23

On 10 Nov 1998 11:07:39 -0500, giff@eng.us.uu.net (Frank Gifford)
wrote:

>>On Sun, 08 Nov 1998 22:09:22 GMT, in &LT;36461506.7272610@news.visi.com>,
>>in sci.crypt schneier@counterpane.com (Bruce Schneier) wrote:
>>
>>>This will become important in the AES selection process.  There are
>>>fifteen candidates.  If an attack is discovered against a candidate,
>>>even if it is a theoretical attack, it will likely knock that cipher
>>>out of the running as long as there are alternatives that prevent such
>>>attacks.
>
>More to the point, it's possible that what is a theoretical attack today
>could be tweaked by others in the crypto community and make it into a
>viable attack for tomorrow.

"Attacks always get better."

Bruce
**********************************************************************
Bruce Schneier, President, Counterpane Systems     Phone: 612-823-1098
101 E Minnehaha Parkway, Minneapolis, MN  55419      Fax: 612-823-1590
           Free crypto newsletter.  See:  http://www.counterpane.com
```

```
Subject: Re: Many attacks, one break?
Date: Tue, 10 Nov 1998 18:44:13 GMT
From: ritter@io.com (Terry Ritter)
Message-ID: &LT;36488971.5055875@news.io.com>
References: &LT;729ocb$esk@perrier.eng.us.uu.net>
Newsgroups: sci.crypt
Lines: 21


On 10 Nov 1998 11:07:39 -0500, in &LT;729ocb$esk@perrier.eng.us.uu.net>,
in sci.crypt giff@eng.us.uu.net (Frank Gifford) wrote:

>[...]
>More to the point, it's possible that what is a theoretical attack today
>could be tweaked by others in the crypto community and make it into a
>viable attack for tomorrow.

Anything is possible already.  We don't need a theoretical attack for
that.

In fact, a theoretical attack could be seen as a clear bill of health
for a cipher with respect to that attack, unless there is some
particular basis for an extrapolation of weakness.

---
Terry Ritter   ritter@io.com   http://www.io.com/~ritter/
Crypto Glossary   http://www.io.com/~ritter/GLOSSARY.HTM
```

```
Subject: Re: Many attacks, one break?
Date: Tue, 10 Nov 1998 18:29:01 GMT
From: schneier@counterpane.com (Bruce Schneier)
Message-ID: &LT;364a85af.2018107@news.visi.com>
References: &LT;3647c322.8278074@news.io.com>
Newsgroups: sci.crypt
Lines: 26

On Tue, 10 Nov 1998 04:38:12 GMT, ritter@io.com (Terry Ritter) wrote:
>I claim that cryptanalysis provides no information at all beyond a
>break or argument of weakness.
>
>A cipher without an academic break is not necessarily stronger, it
>just has yet to be broken.  One could even argue that the broken
>cipher is a better bet because it is better understood.

I just don't agree.  And I am willing to disagree.

>>This will become important in the AES selection process.  There are
>>fifteen candidates.  If an attack is discovered against a candidate,
>>even if it is a theoretical attack, it will likely knock that cipher
>>out of the running as long as there are alternatives that prevent such
>>attacks.
>
>Without a deeper argument of weakness, avoiding a cipher because of an
>impractical attack would seem irrational.

And it is possible that we are all irrational.  Such is life.

Bruce
**********************************************************************
Bruce Schneier, President, Counterpane Systems     Phone: 612-823-1098
101 E Minnehaha Parkway, Minneapolis, MN  55419      Fax: 612-823-1590
           Free crypto newsletter.  See:  http://www.counterpane.com
```

Terry Ritter, his current address, and his top page.

Last updated: 1999-01-19

What's the Meaning of "Invent"?

A Ciphers By Ritter Page

What does it take to be "first to invent," and what does it take to prove it?

Contents

- 1999-01-28 skubi@solsoft.fr: "What's the meaning of 'invent' in the expression 'first to invent'?"
- 1999-01-28 David Kiewit: "The law is set up to keep casual poachers... from using the courts to steal rights from the people who really did the work to make them a success." "The law requires that one be continually diligent during the development period."
- 1999-01-29 John Pederson: "Beautifully put."
- 1999-01-29 Tracy Aquilla: "...'invent' in this context means first to conceive, however the first to conceive can lose a priority contest if she is not first to reduce to practice and is not diligent...." "...diligence is relevant only when an inventor is first to conceive but second to reduce to practice...." "...every event must be independently corroborated by another person...."
- 1999-01-29 David Kiewit: "Another game plan is to forget about these nearly ritualistic documentation practices and get on with completing the invention. File first, and let your prospective opponent get further behind by chasing autographs."
- 1999-01-29 John Pederson: "...during the last 20 years of that time, we decided to adopt exactly this game plan." "Now under the new provisional application rules, I would file immediate automatic PPAs on all invention disclosures...."
- 1999-01-31 Bruce Hayden: "The times I have seen diligence make any difference whatsoever is as a junior party in an interference...."
- 1999-01-31 John Pederson: "Moral of the story -- being senior party gets you 99% of the way home in an interference. And, at least IMO, the remaining 1% chance is just not worth the time, pain and sweat of constant documentation by the technical community and policing of it by the patent professionals."
- 1999-01-31 Bruce Hayden: "In my last employment, we had a couple where we appeared to have the earlier conception date, but destruction of records precluded us proving diligence...."
- 1999-01-31 Ed Suominen: "A PPA is the world's most easily discoverable draft patent application." "Let's not forget the value of priority records for Rule 131 declarations, either."
- 1999-02-01 David Kiewit: "Showing that you were working on the problem before an unpleasant reference was published is certainly one use for priority records."
- 1999-02-01 John Pederson: I continue to feel that as a rule, all too much time and effort is devoted to record-keeping for the amount of benefit at stake, and that a PPA is a new tool...."

Subject: What's the meaning of "invent" in "first to invent" ?
Date: Thu, 28 Jan 1999 22:09:36 GMT
From: skubi@solsoft.fr
Message-ID: <78qn6p$12r$1@nnrp1.dejanews.com>
Newsgroups: misc.int-property
Lines: 30

What's the meaning of "invent" in the expression "first to invent" ?

Is it enough to have the idea, or I must have something more
(like a description of the invented thing sufficient for an
average man skilled in the art to build the thing) ?

In other words: if I claim "a gizmo comprising a shpuntz and a frogifier",
and I want to say (for the purpose of the American patent law, that is)
that I invented this at time T, is it enough to prove that at time T
I had a document containing

 - the sentence "it would be a good idea to build a gizmo comprising
a shpuntz and a frogifier"

 - or the above sentence plus a description of the gizmo (assuming that
there are many ways of arranging the shpuntz and the frogifier, and it
is not obvious which arrangement gives useful results)

 - or the above sentence plus a description of why the gizmo is useful/how it
should be used

 - or all of these

Thank you in advance.

BTW, I like this newsgroup very much. So far, I got very interesting
and friendly advice. Thanks to all.

-----------== Posted via Deja News, The Discussion Network ==----------
http://www.dejanews.com/   Search, Read, Discuss, or Start Your Own

---

Subject: Re: What's the meaning of "invent" in "first to invent" ?
Date: Thu, 28 Jan 1999 19:56:34 -0500
From: David Kiewit <dak@patent-faq.com>
Message-ID: <36B10742.733E5D69@patent-faq.com>
References: <78qn6p$12r$1@nnrp1.dejanews.com>
Newsgroups: misc.int-property
Lines: 54

skubi@solsoft.fr wrote:
>
> What's the meaning of "invent" in the expression "first to invent" ?

I'm glad you asked. Most of the folks who hear the phrase make a totally
wrong assumption about what the answer is.
>
> Is it enough to have the idea, or I must have something more
> (like a description of the invented thing sufficient for an
> average man skilled in the art to build the thing) ?

Someone once said "Success has many fathers, failure is an orphan".
Finding someone who "thought of that a long time and must have been
ripped off by someone he talked to" is easy. The law is set up to keep
casual poachers of that sort from using the courts to steal rights from
the people who really did the work to make them a success.

> In other words: if I claim "a gizmo comprising a shpuntz and a frogifier",
> and I want to say (for the purpose of the American patent law, that is)
> that I invented this at time T, is it enough to prove that at time T
> I had a document containing
>
> - the sentence "it would be a good idea to build a gizmo comprising
> a shpuntz and a frogifier"

No. The law requires that one be continually diligent during the
development period. If the idea is put on the shelf for a while and then
resurrected, the "earliest date" moves up to the time when the project
started again.

Many individual inventors document conception. Few document progress.
Companies with reasonably well managed engineering staffs document
progress up one side and down the other ("How many hours did you charge
to the shpuntz project during the last week of September 1996?" is a
pretty easy question for an employed engineer to answer.

Individual US inventors almost always play the game according to "first
to file" rules in a "first to invent" system. Net, in a priority
dispute, the little guy is the designated loser. But most of the
inventors' groups are the most virulent defenders of first to invent --
a droll situation.

Best regards

Dave

--
David A. Kiewit   (727) 866-0669
5901 3rd St.South, St. Petersburg, FL 33705
Registered Patent Agent   http://patent-faq.com

Subject: Re: What's the meaning of "invent" in "first to invent" ?
Date: Fri, 29 Jan 1999 03:46:55 GMT
From: jptucson@azstarnet.com (John Pederson)
Message-ID: <36b12f0c.1915462@news.azstarnet.com>
References: <36B10742.733E5D69@patent-faq.com>
Newsgroups: misc.int-property
Lines: 53

On Thu, 28 Jan 1999 19:56:34 -0500, David Kiewit <dak@patent-faq.com>
wrote:

>
>
>skubi@solsoft.fr wrote:
>>
>> What's the meaning of "invent" in the expression "first to invent" ?
>
>I'm glad you asked. Most of the folks who hear the phrase make a totally
>wrong assumption about what the answer is.
>
>> Is it enough to have the idea, or I must have something more
>> (like a description of the invented thing sufficient for an
>> average man skilled in the art to build the thing) ?
>
>Someone once said "Success has many fathers, failure is an orphan".
>Finding someone who "thought of that a long time and must have been
>ripped off by someone he talked to" is easy. The law is set up to keep
>casual poachers of that sort from using the courts to steal rights from
>the people who really did the work to make them a success.
>
>> In other words: if I claim "a gizmo comprising a shpuntz and a frogifier",
>> and I want to say (for the purpose of the American patent law, that is)
>> that I invented this at time T, is it enough to prove that at time T
>> I had a document containing
>>
>> - the sentence "it would be a good idea to build a gizmo comprising
>> a shpuntz and a frogifier"
>
>No. The law requires that one be continually diligent during the
>development period. If the idea is put on the shelf for a while and then
>resurrected, the "earliest date" moves up to the time when the project
>started again.
>
>Many individual inventors document conception. Few document progress.
>Companies with reasonably well managed engineering staffs document
>progress up one side and down the other ("How many hours did you charge
>to the shpuntz project during the last week of September 1996?" is a
>pretty easy question for an employed engineer to answer.
>
>Individual US inventors almost always play the game according to "first
>to file" rules in a "first to invent" system. Net, in a priority
>dispute, the little guy is the designated loser. But most of the
>inventors' groups are the most virulent defenders of first to invent --
>a droll situation.

Amen.   Beautifully put.   And so true.

John Pederson, Retired
Former Director of Patents
Major Manufacturing Company

Subject: Re: What's the meaning of "invent" in "first to invent" ?
Date: Fri, 29 Jan 99 15:29:52 GMT
From: taquilla@erols.com (Tracy Aquilla)
Message-ID: <taquilla.1268011432B@news.erols.com>
References: <36B10742.733E5D69@patent-faq.com>
Newsgroups: misc.int-property
Lines: 89

In Article <36B10742.733E5D69@patent-faq.com>, David Kiewit
<dak@patent-faq.com> wrote:
>skubi@solsoft.fr wrote:
>>
>> What's the meaning of "invent" in the expression "first to invent" ?

Since this is a legal term of art, it is not something that is susceptible
to a one or two line definition. Generally, as a starting point, "invent" in
this context means first to conceive, however the first to conceive can lose
a priority contest if she is not first to reduce to practice and is not
diligent (see below). Conception, reduction to practice, and diligence are
also terms of art that have a specific legal meaning.

>I'm glad you asked. Most of the folks who hear the phrase make a totally
>wrong assumption about what the answer is.

Perhaps this is because the priority rules in the US system are a bit
complicated?

>> Is it enough to have the idea, or I must have something more
>> (like a description of the invented thing sufficient for an
>> average man skilled in the art to build the thing) ?
>
>Someone once said "Success has many fathers, failure is an orphan".
>Finding someone who "thought of that a long time and must have been
>ripped off by someone he talked to" is easy. The law is set up to keep
>casual poachers of that sort from using the courts to steal rights from
>the people who really did the work to make them a success.

You mean there are actually people who would do things like that? ;-)

>> In other words: if I claim "a gizmo comprising a shpuntz and a frogifier",
>> and I want to say (for the purpose of the American patent law, that is)
>> that I invented this at time T, is it enough to prove that at time T
>> I had a document containing
>>
>> - the sentence "it would be a good idea to build a gizmo comprising
>> a shpuntz and a frogifier"
>
>No. The law requires that one be continually diligent during the
>development period. If the idea is put on the shelf for a while and then
>resurrected, the "earliest date" moves up to the time when the project
>started again.

Basically true, however, diligence is relevant only when an inventor is
first to conceive but second to reduce to practice (assuming only two-party
interference). The inventor who is first to conceive and last to reduce to
practice is the "first to invent" only if she has been diligent prior to the
time the second inventor enters the field. Thus, the first to conceive may
'put it on the shelf', as long as her diligence is resumed before the second
inventor's conception, and is continuous thereafter until the invention is
reduced to practice. Since filing a US patent application constitutes
constructive reduction to practice, in some cases, merely recording
conception and filing an application may be enough to establish that one was
the "first to invent". However, if the second to conceive can prove an
earlier reduction to practice, diligence of the first to conceive becomes
critical. Proving diligence can be very difficult without good records, and
this has caused serious problems for many independent inventors (eg., Gordon
Gould).

>Many individual inventors document conception. Few document progress.

Yes, this is a common problem. A second common problem for small/individual
inventors is that every event must be independently corroborated by another
person, and all records should be signed, dated and notarized. Small
inventors and those who are new to the game sometimes keep poor records in
this regard, even with respect to conception.

>Companies with reasonably well managed engineering staffs document
>progress up one side and down the other ("How many hours did you charge
>to the shpuntz project during the last week of September 1996?" is a
>pretty easy question for an employed engineer to answer.

Yes, companies that are active in the patent field are generally aware of
the rules and know how to preserve their priority. They also know how to
copy claims and provoke interferences, and can generally afford highly
competent counsel.

>Individual US inventors almost always play the game according to "first
>to file" rules in a "first to invent" system. Net, in a priority
>dispute, the little guy is the designated loser. But most of the
>inventors' groups are the most virulent defenders of first to invent --
>a droll situation.

Yes, this is somewhat ironic.
Tracy

(Note that although it might sound like I think I know what I am talking
about here, this post does not constitute legal advice.)

Subject: Re: What's the meaning of "invent" in "first to invent" ?
Date: Fri, 29 Jan 1999 14:14:35 -0500
From: David Kiewit <dak@patent-faq.com>
Message-ID: <36B2089A.6699AD4@patent-faq.com>
References: <taquilla.1268011432B@news.erols.com>
Newsgroups: misc.int-property
Lines: 41

>
> In Article <36B10742.733E5D69@patent-faq.com>, David Kiewit
> <dak@patent-faq.com> wrote:
<big snip>
> >Many individual inventors document conception. Few document progress.
>
and Tracy Aquilla answered

> Yes, this is a common problem. A second common problem for small/individual
> inventors is that every event must be independently corroborated by another
> person, and all records should be signed, dated and notarized. Small
> inventors and those who are new to the game sometimes keep poor records in
> this regard, even with respect to conception.

Another game plan is to forget about these nearly ritualistic
documentation practices and get on with completing the invention. File
first, and let your prospective opponent get further behind by chasing
autographs.

Having to live with your filing date as your date of invention can be
unpleasant -- if you're not first to file, you lose. But interferences
are rare, so most of the time (98+%??) documenting diligence is a waste
of time. Additionally, the cost of seeing an interference through to the
conclusion is commonly $100K+, with the initial preliminary motion stage
commonly hitting $10K or so. Someplace out there, there's a very
disaffected inventor who spent a lot of time documenting diligence and
then threw all the paper out rather than sustain interference costs.

The alternative isn't for everyone. I, for one, have used it on my own
inventions, where documenting diligence would have been a hassle.

Best regards

Dave
--
David A. Kiewit   (727) 866-0669
5901 3rd St.South, St. Petersburg, FL 33705
Registered Patent Agent   http://patent-faq.com

Subject: Re: What's the meaning of "invent" in "first to invent" ?
Date: Fri, 29 Jan 1999 21:38:15 GMT
From: jptucson@azstarnet.com (John Pederson)
Message-ID: <36b225da.1210867@news.azstarnet.com>
References: <36B2089A.6699AD4@patent-faq.com>
Newsgroups: misc.int-property
Lines: 59

On Fri, 29 Jan 1999 14:14:35 -0500, David Kiewit <dak@patent-faq.com>
wrote:

>
>
>
>>
>> In Article <36B10742.733E5D69@patent-faq.com>, David Kiewit
>> <dak@patent-faq.com> wrote:
><big snip>
>> >Many individual inventors document conception. Few document progress.
>>
>and Tracy Aquilla answered
>
>> Yes, this is a common problem. A second common problem for small/individual
>> inventors is that every event must be independently corroborated by another
>> person, and all records should be signed, dated and notarized. Small
>> inventors and those who are new to the game sometimes keep poor records in
>> this regard, even with respect to conception.
>
>Another game plan is to forget about these nearly ritualistic
>documentation practices and get on with completing the invention. File
>first, and let your prospective opponent get further behind by chasing
>autographs.

Yes!!
>
>Having to live with your filing date as your date of invention can be
>unpleasant -- if you're not first to file, you lose. But interferences
>are rare, so most of the time (98+%??) documenting diligence is a waste
>of time. Additionally, the cost of seeing an interference through to the
>conclusion is commonly $100K+, with the initial preliminary motion stage
>commonly hitting $10K or so. Someplace out there, there's a very
>disaffected inventor who spent a lot of time documenting diligence and
>then threw all the paper out rather than sustain interference costs.

Yes!!

To the best of my recollection, in the 43 years I spent as a patent
professional and, ultimately, as Director of Patents with a major US
manufacturer in a patent active industry, diligence or lack of
diligence never made a difference in any interference or other case in
which we were involved.

Actually, during the last 20 years of that time, we decided to adopt
exactly this game plan.   And the time saved by our technical
community by not documenting diligence was used instead to make
additional inventions and/or otherwise improve productivity.

Now under the new provisional application rules, I would file
immediate automatic PPAs on all invention disclosures as received by
my patent department, thus nailing down the earliest possible filing
date for those on which we decided to file complete applications.
This would give the best of all worlds, for only $150 per PPA.

John Pederson, Retired
Former Director of Patents
Major Manufacturing Company

Subject: Re: What's the meaning of "invent" in "first to invent" ?
Date: Sun, 31 Jan 1999 08:54:24 -0700
From: Bruce Hayden <bhayden@ieee.org>
Message-ID: <36B47CB0.DDB3434D@ieee.org>
References: <36b225da.1210867@news.azstarnet.com>
Newsgroups: misc.int-property
Lines: 33

John Pederson wrote:
> >
> >Having to live with your filing date as your date of invention can be
> >unpleasant -- if you're not first to file, you lose. But interferences
> >are rare, so most of the time (98+%??) documenting diligence is a waste
> >of time. Additionally, the cost of seeing an interference through to the
> >conclusion is commonly $100K+, with the initial preliminary motion stage
> >commonly hitting $10K or so. Someplace out there, there's a very
> >disaffected inventor who spent a lot of time documenting diligence and
> >then threw all the paper out rather than sustain interference costs.
>
> Yes!!
>
> To the best of my recollection, in the 43 years I spent as a patent
> professional and, ultimately, as Director of Patents with a major US
> manufacturer in a patent active industry, diligence or lack of
> diligence never made a difference in any interference or other case in
> which we were involved.

The times I have seen diligence make any difference whatsoever is
as a junior party in an interference - which actually reinforces
your point of filing early, and hopefully becoming the senior
party if an interference is called.
--

The preceding was not a legal opinion, and is not my employer's.
Original portions Copyright 1999 Bruce E. Hayden,all rights reserved
My work may be copied in whole or part, with proper attribution,
as long as the copying is not for commercial gain.

Bruce E. Hayden   bhayden@acm.org
Phoenix, Arizona   bhayden@ieee.org
                                     bhayden@copatlaw.com

Subject: Re: What's the meaning of "invent" in "first to invent" ?
Date: Sun, 31 Jan 1999 17:07:08 GMT
From: jptucson@azstarnet.com (John Pederson)
Message-ID: <36b5887c.4677942@news.azstarnet.com>
References: <36B47CB0.DDB3434D@ieee.org>
Newsgroups: misc.int-property
Lines: 50

On Sun, 31 Jan 1999 08:54:24 -0700, Bruce Hayden <bhayden@ieee.org>
wrote:

>John Pederson wrote:
>> >
>> >Having to live with your filing date as your date of invention can be
>> >unpleasant -- if you're not first to file, you lose. But interferences
>> >are rare, so most of the time (98+%??) documenting diligence is a waste
>> >of time. Additionally, the cost of seeing an interference through to the
>> >conclusion is commonly $100K+, with the initial preliminary motion stage
>> >commonly hitting $10K or so. Someplace out there, there's a very
>> >disaffected inventor who spent a lot of time documenting diligence and
>> >then threw all the paper out rather than sustain interference costs.
>>
>> Yes!!
>>
>> To the best of my recollection, in the 43 years I spent as a patent
>> professional and, ultimately, as Director of Patents with a major US
>> manufacturer in a patent active industry, diligence or lack of
>> diligence never made a difference in any interference or other case in
>> which we were involved.
>
>The times I have seen diligence make any difference whatsoever is
>as a junior party in an interference - which actually reinforces
>your point of filing early, and hopefully becoming the senior
>party if an interference is called.

Sure.   My point was that on the few occasions when we were junior
party, we could not have been saved by proof of diligence (no matter
how complete) because it turned out that the senior party was first to
conceive as well as being first to reduce to practice by virtue of
being first to file.   So even when we turned out to be junior
party, our efforts at documenting diligence went for naught.   Our
batting average for all our efforts to maintain diligence records was
.000

BTW, at least we saved the pain and expense of finding this out the
hard way, because we worked out settlement agreements resolving the
priority issue by informal exchange of proofs.

Moral of the story -- being senior party gets you 99% of the way home
in an interference.   And, at least IMO, the remaining 1% chance is
just not worth the time, pain and sweat of constant documentation by
the technical community and policing of it by the patent
professionals.

John Pederson, Retired
Former Director of Patents
Major Manufacturing Company

Subject: Re: What's the meaning of "invent" in "first to invent" ?
Date: Sun, 31 Jan 1999 14:12:54 -0700
From: Bruce Hayden <bhayden@ieee.org>
Message-ID: <36B4C756.A316C1F3@ieee.org>
References: <36b5887c.4677942@news.azstarnet.com>
Newsgroups: misc.int-property
Lines: 47

John Pederson wrote:

> >
> >The times I have seen diligence make any difference whatsoever is
> >as a junior party in an interference - which actually reinforces
> >your point of filing early, and hopefully becoming the senior
> >party if an interference is called.
>
> Sure.   My point was that on the few occasions when we were junior
> party, we could not have been saved by proof of diligence (no matter
> how complete) because it turned out that the senior party was first to
> conceive as well as being first to reduce to practice by virtue of
> being first to file.   So even when we turned out to be junior
> party, our efforts at documenting diligence went for naught.   Our
> batting average for all our efforts to maintain diligence records was
> .000

In my last employment, we had a couple where we appeared to have the
earlier conception date, but destruction of records precluded us
proving diligence (or probably more likely there was a gap).
So, as junior party, we were forced to cave.

> BTW, at least we saved the pain and expense of finding this out the
> hard way, because we worked out settlement agreements resolving the
> priority issue by informal exchange of proofs.

Similar.

> Moral of the story -- being senior party gets you 99% of the way home
> in an interference.   And, at least IMO, the remaining 1% chance is
> just not worth the time, pain and sweat of constant documentation by
> the technical community and policing of it by the patent
> professionals.

Obviously my "exception" proves your rule since if we had been
senior party in any of those situations, it would have been up
to the junior party to show diligence instead of us.
--

The preceding was not a legal opinion, and is not my employer's.
Original portions Copyright 1999 Bruce E. Hayden,all rights reserved
My work may be copied in whole or part, with proper attribution,
as long as the copying is not for commercial gain.

Bruce E. Hayden   bhayden@acm.org
Phoenix, Arizona   bhayden@ieee.org
                                     bhayden@copatlaw.com

Subject: Re: What's the meaning of "invent" in "first to invent" ?
Date: Sun, 31 Jan 1999 21:30:47 -0700
From: Ed Suominen <see-my@web-site.com>
Message-ID: <36B52DF7.28DD59A@web-site.com>
References: <36b225da.1210867@news.azstarnet.com>
Newsgroups: misc.int-property
Lines: 36

John Pederson wrote:

> Now under the new provisional application rules, I would file
> immediate automatic PPAs on all invention disclosures as received by
> my patent department, thus nailing down the earliest possible filing
> date for those on which we decided to file complete applications.
> This would give the best of all worlds, for only $150 per PPA.
>

John, I respect your experience and history of well-written posts. I just want to
make one cautionary note.

A PPA is the world's most easily discoverable draft patent application. Inventors
can write narrowing and unguarded statements in a PPA such as, "the other piece of
code in Appendix B is probably obvious - everyone here thinks so," and "process A
only works with blue widgets. It isn't suitable at all for use with red widgets."
Do you really want to put such statements in the file wrapper with an "automatic"
PPA? If you go through the disclosure to "clean it up," you probably will be
spending enough time to justify just getting started on the claims and
specification.

Let's not forget the value of priority records for Rule 131 declarations, either.
I suspect more patents are issued under Rule 131 than after interferences. Even in
the unlikely event of an interference, you might wind up being more likely to win
good patent protection by using priority records than by relying on a narrowly
written provisional application with lots of legal holes in it.

--
Nothing in this message is to be construed as legal
advice, or the opinion of my firm or its clients.

Ed Suominen, Patent Agent
Squire, Sanders & Dempsey L.L.P.
Web Site: http://eepatents.com

Subject: Re: What's the meaning of "invent" in "first to invent" ?
Date: Mon, 01 Feb 1999 06:21:04 -0500
From: David Kiewit <dak@patent-faq.com>
Message-ID: <36B58E20.9396A5AF@patent-faq.com>
References: <36B52DF7.28DD59A@web-site.com>
Newsgroups: misc.int-property
Lines: 21

Ed Suominen wrote:
>
> Let's not forget the value of priority records for Rule 131 declarations, either.
> I suspect more patents are issued under Rule 131 than after interferences.

Showing that you were working on the problem before an unpleasant
reference was published is certainly one use for priority records. But
Rule 131 diligence is a much softer, fuzzier animal than interference
diligence.

Best regards

Dave

--
David A. Kiewit   (727) 866-0669
5901 3rd St.South, St. Petersburg, FL 33705
Registered Patent Agent   http://patent-faq.com

Subject: Re: What's the meaning of "invent" in "first to invent" ?
Date: Mon, 01 Feb 1999 16:35:00 GMT
From: jptucson@azstarnet.com (John Pederson)
Message-ID: <36b5c7ae.891160@news.azstarnet.com>
References: <36B52DF7.28DD59A@web-site.com>
Newsgroups: misc.int-property
Lines: 105

On Sun, 31 Jan 1999 21:30:47 -0700, Ed Suominen <see-my@web-site.com>
wrote:

>John Pederson wrote:
>
>> Now under the new provisional application rules, I would file
>> immediate automatic PPAs on all invention disclosures as received by
>> my patent department, thus nailing down the earliest possible filing
>> date for those on which we decided to file complete applications.
>> This would give the best of all worlds, for only $150 per PPA.
>>
>
>John, I respect your experience and history of well-written posts. I just want to
>make one cautionary note.
>
>A PPA is the world's most easily discoverable draft patent application.

True.   Good point.

>Inventors
>can write narrowing and unguarded statements in a PPA such as, "the other piece of
>code in Appendix B is probably obvious - everyone here thinks so," and "process A
>only works with blue widgets. It isn't suitable at all for use with red widgets."

Also true, and lamentable.   Of course, we worked closely with our
technical community and tried hard to teach them not to do such
things, sometimes with success and sometimes not.   In that regard,
it's worth noting that we were a multi-product company, with all
engineering, advanced development and research, and patent department,
centralized in one metro area.   Also our patent department reported
to top management in the early years and to the law department in
later years, so we had more credibility and I suspect more success in
such efforts than if we had been subservient to Engineering
management.

>Do you really want to put such statements in the file wrapper with an "automatic"
>PPA? If you go through the disclosure to "clean it up," you probably will be
>spending enough time to justify just getting started on the claims and
>specification.

Going through to "clean it up" would be a good idea, and would not be
so time consuming in this day and age.   Just scan it in, bring it
up on a word processor, read and edit as you go, and print out.
Shouldn't take more than a few minutes a page.
>
>Let's not forget the value of priority records for Rule 131 declarations, either.
>I suspect more patents are issued under Rule 131 than after interferences. Even in
>the unlikely event of an interference, you might wind up being more likely to win
>good patent protection by using priority records than by relying on a narrowly
>written provisional application with lots of legal holes in it.

Again true.   And another good point.   But I know of nothing that
says we can't use the priority records for Rule 131 even if we have
already filed the (possibly "cleaned up") invention disclosure as a
PPA.   And my experience was that we used Rule 131 in very few cases
also -- the frequency was more often than interference situations, but
still definitely under a couple of percent of the cases.

A couple of additional comments:

Nothing says that we have to claim the PPA filing date even in a
complete application filed within a year.   So there is an
opportunity to "abort" the PPA (by not referencing it and not claiming
its priority date) at the time of filing the complete, if it is an
important case and it is thought that it would be better to rely on
priority documents only.   BTW, we followed the practice of
destroying all drafts at the time of filing a complete application,
and that practice could be extended to destruction also of our copies
of an "aborted" PPA.

Sure the PPA could still be discoverable in contested litigation -- so
are the corresponding "uncleaned up" priority documents.   And sure
we might wind up once a decade or so losing a case that we might
otherwise have won.   It's a risk/reward evaluation, and you can't
win 'em all.   BTW, we haven't been winning 'em all by doing things
the old way.

Moreover the risk/reward weightings may vary from one situation to
another.   Our industry was a patent active industry, and there were
many blanket cross licenses between the major players.   As
pointed out above, we were a multi-product centralized company.
And we had enough numbers that the statistical odds were meaningful.
For those without the numbers, or in single-product companies, or
companies without patent departments, or companies with widely
decentralized technical and patent operations, or for private
inventors, the outlook might be different.   Then again, it might
not.

 I continue to feel that as a rule, all too much time and effort is
devoted to record-keeping for the amount of benefit at stake, and that
a PPA is a new tool that can be used not only to buy time for finding
out if the invention is a winner before spending the much bigger bucks
to go after a patent, but also to reduce or eliminate the chances of
being victimized by the starry-eyed and widely held misconception that
we have a true first-to-invent patent system.

Thanks for the observations and comments.   Any reactions to mine
above?

-

John Pederson, Retired
Former Director of Patents
Major Manufacturing Company

Terry Ritter, his current address, and his top page.

Last updated: 1999-02-20

Patent Notebook Consequences

A *Ciphers By Ritter* Page

Needing to fill in a lab notebook -- and get witness signatures -- is a hassle which leads to omissions. Can we avoid this?

Contents

- 1998-09-09 Jesse Salb: "...the standard, handwritten, witnessed lab notebook model seems inconvenient and outmoded. Is there an acceptable alternative?"
- 1998-09-10 David D. Lowry: "The witnessed and signed bound lab notebook is outmoded and inconvenient, but it is still the best way to record your ideas and data."
- 1998-09-10 Gideon Gimlan: "There is no magic to a bound notebook. The magic lies in having some person testify on your behalf and what that testimony proves."
- 1998-09-11 Carl Oppedahl: "After an adequate provisional application has been filed on a particular invention, then as to that invention you can stop worrying about 'inconvenient, outmoded' handwritten lab notebook."
- 1998-09-11 Bigipguy: "...under the law, any evidence of conception and diligence must be corroborated. If it's not corroborated, you might as well forget it. And the date of corroboration is what controls. In the context of a laboratory notebook, that means the date the notebook is witnessed BY A NON-INVENTOR, not the date an inventor writes down."

Subject: Lab notebook documentation
Date: Wed, 09 Sep 1998 11:57:56 -0700
From: jsalb@ix.netcom.com (Jesse Salb)
Message-ID: <35f7cf68.218635090@nntp.ix.netcom.com>
Newsgroups: misc.int-property
Lines: 8

Can someone offer an opinion on, or point me toward a web discussion
of, the acceptable forms of documentation for new inventions. I do
mostly computational chemistry, in which so much of the work and
results are computer-based, that the standard, handwritten, witnessed
lab notebook model seems inconvenient and outmoded. Is there an
acceptable alternative?

Jesse

Subject: Re: Lab notebook documentation
Date: Thu, 10 Sep 1998 09:13:03 -0400
From: "David D. Lowry" <ddlNOSPAM@dbrc.com>
Message-ID: <35F7D05F.1A19@dbrc.com>
References: <35f7cf68.218635090@nntp.ix.netcom.com>
Newsgroups: misc.int-property
Lines: 36

Jesse Salb wrote:
>
> Can someone offer an opinion on, or point me toward a web discussion
> of, the acceptable forms of documentation for new inventions. I do
> mostly computational chemistry, in which so much of the work and
> results are computer-based, that the standard, handwritten, witnessed
> lab notebook model seems inconvenient and outmoded. Is there an
> acceptable alternative?
>
> Jesse

The witnessed and signed bound lab notebook is outmoded and
inconvenient, but it is still the best way to record your ideas and
data.

I've heard of some companies selling "electronic lab notebooks", which
are applications that run on PCs, and strive to record and date your
entries. How well they work I don't know. The problem which they all
must overcome is recording in such a way that editing the content or the
date is impossible. Obviously this can be done, for example using WORM
(write once, read many) CDs, or various encryption techniques. But the
other problem is that if a dispute ever gets to court, the data will be
disputed. Obviously real lab notebooks can altered, but they are well
accepted as evidence, while the newer electronic systems are not
"proven" yet.

Stick to the lab notebook. It sucks to keep adding entries, but you
will find it useful beyond the usual legal requirements.

DDL
[David D. Lowry (remove "NOSPAM" to email)
[Registered Patent Attorney ddl@NOSPAMdbrc.com
[Dike, Bronstein, Roberts & Cushman LLP www.dbrc.com
[Boston, MA 02109 617 523-3400
[
[CLOS >= java*10 > C++ * 100

Subject: Re: Lab notebook documentation
Date: Thu, 10 Sep 1998 21:18:11 -0700
From: Gideon Gimlan <gimlan@earthlink.net>
Message-ID: <35F8A483.CBEE908F@earthlink.net>
References: <35F7D05F.1A19@dbrc.com>
Newsgroups: misc.int-property
Lines: 53

First, you must understand WHY you would want to document your date of earliest
invention. It is only out of worry that you will get into a contest as to who was
first to invent in the United States.The documentation does not secure any
patent rights for you. You have to file a patent application for that.

Second, you must understand that the question becomes whether you have
independent corrobaritive evidence if you ever do get into an interference
fight (who was first to invent). There is no magic to a bound notebook. The
magic lies in having some person testify on your behalf and what that
testimony proves. If your notebook is not witnessed by a noninventor, it is basically
worthless

David D. Lowry wrote:

> Jesse Salb wrote:
> >
> > Can someone offer an opinion on, or point me toward a web discussion
> > of, the acceptable forms of documentation for new inventions. I do
> > mostly computational chemistry, in which so much of the work and
> > results are computer-based, that the standard, handwritten, witnessed
> > lab notebook model seems inconvenient and outmoded. Is there an
> > acceptable alternative?
> >
> > Jesse
>
> The witnessed and signed bound lab notebook is outmoded and
> inconvenient, but it is still the best way to record your ideas and
> data.
>
> I've heard of some companies selling "electronic lab notebooks", which
> are applications that run on PCs, and strive to record and date your
> entries. How well they work I don't know. The problem which they all
> must overcome is recording in such a way that editing the content or the
> date is impossible. Obviously this can be done, for example using WORM
> (write once, read many) CDs, or various encryption techniques. But the
> other problem is that if a dispute ever gets to court, the data will be
> disputed. Obviously real lab notebooks can altered, but they are well
> accepted as evidence, while the newer electronic systems are not
> "proven" yet.
>
> Stick to the lab notebook. It sucks to keep adding entries, but you
> will find it useful beyond the usual legal requirements.
>
> DDL
> [David D. Lowry (remove "NOSPAM" to email)
> [Registered Patent Attorney ddl@NOSPAMdbrc.com
> [Dike, Bronstein, Roberts & Cushman LLP www.dbrc.com
> [Boston, MA 02109 617 523-3400
> [
> [CLOS >= java*10 > C++ * 100

Subject: Re: Lab notebook documentation
Date: Fri, 11 Sep 1998 09:06:33 GMT
From: oppedahl@patents.com (Carl Oppedahl)
Message-ID: <6tap6p$nnm192_002@news.panix.com>
References: <35f7cf68.218635090@nntp.ix.netcom.com>
Newsgroups: misc.int-property
Lines: 21

In article <35f7cf68.218635090@nntp.ix.netcom.com>, jsalb@ix.netcom.com (Jesse
Salb) wrote:

>Can someone offer an opinion on, or point me toward a web discussion
>of, the acceptable forms of documentation for new inventions. I do
>mostly computational chemistry, in which so much of the work and
>results are computer-based, that the standard, handwritten, witnessed
>lab notebook model seems inconvenient and outmoded. Is there an
>acceptable alternative?

You didn't say which country you'd like to get patent protection in. If it's
the United States, then you should simply file a provisional patent
application the same day that any of your inventors writes down any new
invention. After an adequate provisional application has been filed on a
particular invention, then as to that invention you can stop worrying about
"inconvenient, outmoded" handwritten lab notebook. The cost, $75 or $150, is
probably insignificant compared with the cost of operating your laboratory.

To read more about provisional patent applications, see
http://www.patents.com/patents.sht .

Subject: Re: Lab notebook documentation
Date: 11 Sep 1998 22:52:51 GMT
From: bigipguy@aol.com (Bigipguy)
Message-ID: <1998091122525100.SAA08160@ladder01.news.aol.com>
References: <6tap6p$nnm192_002@news.panix.com>
Newsgroups: misc.int-property
Lines: 51

In article <6tap6p$nnm192_002@news.panix.com>, oppedahl@patents.com (Carl
Oppedahl) writes:

>You didn't say which country you'd like to get patent protection in. If it's
>the United States, then you should simply file a provisional patent
>application the same day that any of your inventors writes down any new
>invention. After an adequate provisional application has been filed on a
>particular invention, then as to that invention you can stop worrying about
>"inconvenient, outmoded" handwritten lab notebook. The cost, $75 or $150, is
>probably insignificant compared with the cost of operating your laboratory.

This is not good advice.

If you instantly develop a complete invention in a single day, then maybe it
does make sense to file a provisional case that day, I don't know. Though it
will take a lot more than $75 to see that it's done correctly. I'd never want
a non-lawyer to submit statements about what his or her invention is to the PTO
- statements written by non-lawyers are invariably too narrow & may result in
estoppels later. To see that the provisional application is properly written
(and indeed filed), a lawyer should prepare the application. The idea that a
provisional application can be a cheap form of "priority protection" is
illusory - you get what you pay for.

But that assumes that the act of invention is instantaneous. In real research
labs, the reduction to practice does not take place immediately after
conception, but instead may take place over a long period of time. There
likely will be many false starts, and the inventors generally will collect a
lot of data. In a patent interference or in a priority contest in a lawsuit,
evidence of such work may be used to demonstrate both conception and diligence
in reduction to practice. Indeed, without such evidence, the inventor will not
be able to prove conception, nor diligence in reduction to practice.

Moreover, under the law, any evidence of conception and diligence must be
corroborated. If it's not corroborated, you might as well forget it. And the
date of corroboration is what controls. In the context of a laboratory
notebook, that means the date the notebook is witnessed BY A NON-INVENTOR, not
the date an inventor writes down.

Given this, one should not try to rely on provisional filings to preserve
priority of invention. Simply put, it's neither practical nor desirable to try
to file a provisional application to document each day of research. If you
wait until you've got a defined "invention" to start making a record, then
you're vulnerable to a priority attack from someone who has kept adequate
records all along.

Nor is it desirable to dispense with the lab notebook system. The law requires
corroboration of the acts leading to invention, period. Like it or not, if you
don't have a witnessed lab notebook, you probably don't have corroborating
evidence of what you've done. The notebook system may seem outmoded or
anachronistic, but I know of no other record-keeping system that will suffice
for your purposes.

Terry Ritter, his *current address*, and his *top page*.

Last updated: 1999-02-20

Software Patents?

A *Ciphers By Ritter* Page

A discussion of software intellectual property, inevitably drifting into a discussion of software patents, and their impact on free or "open" software.

Contents

- 1998-08-04 Karsten M. Self: "Pamela Samuelson is a highly respected leading, if controversial, intellectual property (IP) scholar who has criticized existing copyright and patent protection of software. She argues for sui generis -- special case -- IP protections for software...."
- 1998-08-4 Roger Schlafly: "...you should realize that her views are way out on the fringe, legally."
- 1998-08-05 Bruce Hayden: "...my special criticism though is directed towards her position on software patents. If you read at least the older versions of her Manifesto, I think that you can best describe her position as not having a clue as to what is patentable and what is not as far as software is concerned."
- 1998-08-05 Karsten M. Self: "Having started with her earlier paper... which challenged the legal arguments for covering software under copyright, I'd gathered she generally felt that SW IP was headed for overprotection, and needs its own form of IP to correct this. The Manifesto makes the opposite case, a decade later...."
- 1998-08-5 Roger Schlafly: "The argument for sui generis seems to depend on the claim that software is qualitatively different from other technologies." "But if this justifies sui gen, then it seems much easier to make the case that other technologies are different, and should have sui gen."
- 1998-08-06 Karsten M. Self: "IC *got* sui generis protection with the SCPA (chip mask copyrighting statute, 17 U.S.C. 409)."
- 1998-08-05 Bruce Hayden: "...the example used for sui generis is semiconductor mask protection. It was obsolete almost from the date passed."
- 1998-08-07 Karsten M. Self: "SW doesn't fit copyright because copyright is designed to control publication while disseminating knowledge. SW uses copyright to control appropriation while disseminating product."
- 1998-08-7 Roger Schlafly: "And do movies, music records, and artwork fit copyright? This distinction seems bizarre to me."
- 1998-08-8 Roger Schlafly: "A sculptor doesn't necessarily reveal his techniques, nor is it obvious from the sculpture. Likewise for many other works."
- 1998-08-08 Karsten M. Self: "...if object code is published and distributed, why should copyright protection cover anything other than the *expression* of technique under software."
- 1998-08-09 Karsten M. Self: "Lotus v. Borland, Sega v. Accolade, and CA v. Altai have significantly weakened the level of protection afforded by copyright. Borland said that program elements may be duplicated without infringement, Accolade that reverse engineering is allowable under Fair Use, and Altai that AFC was the means for seperating copyrightable expression from non-copyrightable ideas in a program."
- 1998-08-09 Andrew C. Greenberg: "...the scope of non-literal infringement has diminished dramatically over the past twenty years. Indeed, it has diminished in some jurisdictions to practically nothing."
- 1998-08-09 Andrew C. Greenberg:
- 1998-08-08 Andrew C. Greenberg: "Many works are perceived not by the expression directly, but by the "execution" of the expression. Records and tape recordings, written music, written dance recordations, scripts, screenplays." "...we need to consider whether the argument being made is: (1) patents are bad; (2) software patents are bad; or (3) bad software patents are bad."
- 1998-08-08 Karsten M. Self: "If anything, the traditional area of copyright seems to be following software into the electronic age." "I'm not buying the instantiation of the program, I'm buying the object code which makes the instantiation possible."
- 1998-08-09 Andrew C. Greenberg: "Different programs... typically have different executions. Sometimes they make a difference, and sometimes they don't." "But it is the same with music, drama and dance."
- 1998-08-05 Steve Peltz: "...I would claim that BOTH patents were 'obvious' (given the current state of the art at the time), almost trivial in fact, and not worthy of patent protection...."
- 1998-08-06 Andrew C. Greenberg: "A disclosure needn't be valuable to justify a patent, merely containing an enabling disclosure of a new, useful and unobvious."
- 1998-08-10 johnny_squire@hotmail.com: "That's the current state of the practice. Is it right, good or constitutionally justified? No."
- 1998-08-10 Karsten M. Self: "We don't need to show that patent claims are being read by other inventors (and they are), but that a patent claim provides security for the inventor to discuss his own invention without fear of losing rights to the idea."
- 1998-08-10 Andrew C. Greenberg: "It's a question of fact whether a patent discloses new, useful and unobvious subject matter. If that is true, then it is apparent it must teach SOMETHING...."
- 1998-08-12 Bruce Hayden: "...pretty much all technologies are thrown in together when determining whether or not the beneifits of the patent system outweigh the costs."
- 1998-08-13 Bruce Hayden: "Allapat basically said that a programmed general purpose computer becomes a special purpose computer, which is a machine under section 101."
- 1998-08-14 Roger Schlafly: "Every change that Congress proposes for copyrights and patents is bad. I see no hope that Congress could produce a sui generis software protection system which is better than what we have."
- 1998-08-14 Andrew C. Greenberg: "Current experiences in patent reform and the database legislation are, well, scary reminders of what sausage legislation can be. Additionally, changes to legislation for inventions rlated to software raise a host of serious consequenes and problems, not the lest of which is the decade of uncertainty as the new laws are construed by the courts."
- 1998-08-14 Andrew C. Greenberg: "Seriously, Johnny, what is your point? That because YOU don't read patents, the patent system is unconstitutional? Nonsense. It is certainly not true that NOBODY reads patents."
- 1998-08-14 johnny_squire@hotmail.com: "Software patents today are problematic (not unconstitutional) because NOBODY OF ORDINARY SKILL IN THE ART reads them."
- 1998-08-17 Roger Schlafly:
- 1998-08-17 Bruce Hayden: "This is what I call wishful thinking law. You wish that software wasn't patentable, and thus it really isn't, despite the courts consistently deciding to the contrary."
- 1998-08-07 Chris Mikkelson: "While re-discovering J-l Gailly's page..., I found a link to the comp.compression FAQ. According to its section on compression patents, run-length encoding is patented!" "I would certainly have suspected that to fail the 'not obvious' requirement."
- 1998-08-7 Roger Schlafly: "Most significant new software products take 2+ years to develop, 6+ months to test, and millions of dollars to bring to market."
- 1998-08-07 Andrew C. Greenberg: "Mathematical or not, the APPLICATION of an algorithm to a particular patent, expressed as an apparatus, article of manufacture or method will be patentable. _See_ State Street Bank."
- 1998-08-10 Craig Burley: "...software patents are unlikely to be anything but a substantial hindrance to the free-software development paradigm...."
- 1998-08-10 Roger Schlafly: "...software patents raise the barrier to entry in the software market, and that fact works to Microsoft's favor."
- 1998-08-11 Craig Burley: "...individuals like myself *cannot afford* to do our volunteer development under an ever-increasing threat of litigation over software patents."
- 1998-08-12 Andrew C. Greenberg: "Legitimate assertion of "good" software patents, it seems to me, is a reasonable result of the patent system. How to deal with it? Engineering around the patent is the most cost-effective solution...." "Software patents are a reality. Deal with it. Undertand them. Know what you are truly getting into, and deal with the consequences."
- 1998-08-12 Karsten M. Self: "OSS cannot wait for the time it would take legal mechanisms to work, and should exploit commercial interests to attain the protection it seeks."
- 1998-08-13 Craig Burley: "I've finally found something in my archives, in a post from Terry Ritter circa September 1991, following up one of my posts, in a thread discussing software patents."
- 1998-08-13 Terry Ritter: "I see quality is a contract between producer and user, delivery for a price.... And when the economic situation is set up so that contract *cannot* be enforced... I think society needs a different solution." "Both abstractly and in practice software *is* hardware: When 'software' runs, there is no software there, just bits and gates and cells and voltages."
- 1998-08-14 Karsten M. Self: "...it's usually the attackers of the status quo who are most vocal -- those who benefit by it are too busy reaping their rewards."
- 1998-08-12 Andrew C. Greenberg: "I am not sure how society benefits by providing that a person who has a legitimate patent covering an invention that can implemented in software benefits by letting others knock off the product and giving it away for free." "...nations with strong IP have been substantially more inventive and substantially stronger in most meaningful areas of technology than those which have not."
- 1998-08-12 Andrew C. Greenberg: "You cannot patent a formula, pure algorithm or law of nature, per se. However, you CAN patent the application of such a formula, algorithm or law of nature

to a concrete and meaningful application, when manifest as a concrete process or apparatus."

- 1998-08-13 PILCH Hartmut: "Software patents are only applicable to 'products', not to 'works'. Whatever is placed under Open Source license terms is only a work (abstract system of ideas) and can never become a product (concrete process or apparatus)."
- 1998-08-13 Andrew C. Greenberg: "Claims are typically directed to apparatus and methods, and not to books."
- 1998-08-13 Bruce Hayden: "I load software into a general purpose computer, and it becomes a special purpose computer. That is usually a machine under the patent laws. What you say in your Open Source license is irrelevant."
- 1998-08-13 Stefaan A Eeckels: "...a programs is basically the stuff that adapts a non-specific tool or machine to a particular problem. Mutatis mutandis, this is similar to the process that adapts a block of metal or wood to perform a particular function."
- 1998-08-13 Craig Burley: "...as I've said, all I've really seen are negatives, and no positives."
- 1998-08-13 Bruce Hayden: "One of the things that you have to fight with examiners all the time over is their human tendency to look at your problem definition, and from that jump to your conclusion. This is not allowed, if the problem definition is itself novel and nonobviousness."
- 1998-08-13 Bruce Hayden:
- 1998-08-13 Craig Burley: "...Newton actually *did* keep his Calculus secret for years, using it for his own purposes, until he realized somebody else was about to publish similar stuff.... Needless to say, this is an argument in *favor* of patenting entire fields of mathematics, or at least algorithms."
- 1998-08-21 Karsten M. Self: "I'd like to create a system which creates very strong incentives to share. But I'd like to leave the option of not sharing available, however costly and/or income-forgoing it might be."
- 1998-08-12 Steve Peltz: "The problem with a lot of software patents, as I see it, is that the solution is obvious once you decide to solve the problem."
- 1998-08-12 Andrew C. Greenberg: "Deciding to define and solve a problem can in and of itself be sufficient to make the solution unobvious."
- 1998-08-13 Joe Buck: "If the publication of the patent is of no value to anyone (because any skilled practitioner assigned the problem would come up with the same solution, and because formulating the problem is of no value either, because it is simply "automate some existing task"), the moral justification for the patent (the one in the US Constitution) doesn't apply."
- 1998-08-21 Tim Smith: "For concrete discussion, here are software patents that seem obvious to me."
- 1998-08-21 Andrew C. Greenberg:
- 1998-08-13 Craig Burley: "It's really silly to waste any time trying to profit off of a few software patents these days -- one can make so much more money doing Real Work."

```
Subject: Pamela Samuelson panelist at Open Source town meeting 8/21/98 San Jose, CA
Date: Tue, 04 Aug 1998 17:58:04 +0000
From: "Karsten M. Self" &LT;kmself@ix.netcom.com>
Message-ID: &LT;35C74BAC.547245AC@ix.netcom.com>
Newsgroups:
misc.int-property,misc.legal.computing,comp.os.linux.misc,comp.os.linux.misc,gnu.misc.discuss
Lines: 26

Information at http://opensource.oreilly.com/townmeet.html

Pamela Samuelson is a highly respected leading, if controversial,
intellectual property (IP) scholar who has criticized existing copyright
and patent protection of software.  She argues for sui generis --
special case -- IP protections for software and is the author of several
significant articles on this topic.  She has a website with
bibliographic and biographic information, as well as a selection of
papers at http://www.sims.berkeley.edu/~pam/

I'm thrilled to see her included as a panelist in the upcoming Open
Source Town Meeting, Friday, August 21, 5-6:30 p.m. at the Fairmont
Hotel in San Jose, California ($10 admission).

Kudos to O'Reilly and/or responsible parties for arranging this.

--
Karsten M. Self (kmself@ix.netcom.com)

    What part of "gestalt" don't you understand?
    Welchen Teil von "gestalt" verstehen Sie nicht?

web:       http://www.netcom.com/~kmself
SAS/Linux: http://www.netcom.com/~kmself/SAS/SAS4Linux.html

 10:41am  up 52 days,  8:10,  3 users,  load average: 1.12, 1.14, 1.17
```

```
Subject: Re: Pamela Samuelson panelist at Open Source town meeting 8/21/98 San Jose,
CA
Date: Tue, 4 Aug 1998 21:36:49 -0700
From: "Roger Schlafly" &LT;nospam.schlafly@cruzio.com>
Message-ID: &LT;6q8nc2$8n8$1@camel25.mindspring.com>
References: &LT;35C74BAC.547245AC@ix.netcom.com>
Newsgroups:
misc.int-property,misc.legal.computing,comp.os.linux.misc,comp.os.linux.misc,gnu.misc.discuss
Lines: 15


Karsten M. Self wrote in message &LT;35C74BAC.547245AC@ix.netcom.com>...
>Information at http://opensource.oreilly.com/townmeet.html
>
>Pamela Samuelson is a highly respected leading, if controversial,
>intellectual property (IP) scholar who has criticized existing copyright
>and patent protection of software.  She argues for sui generis --

If you attend, you should realize that her views are way out on
the fringe, legally.  She has written articles against all patent
protection of software, and against all copyright protection
of object code.
```

Subject: Re: Pamela Samuelson panelist at Open Source town meeting 8/21/98 San Jose, CA
Date: Wed, 05 Aug 1998 00:43:46 -0700
From: Bruce Hayden <bhayden@uswest.net>
Message-ID: <35C80D32.C7F5795F@uswest.net>
References: <6q8nc2$8n8$1@camel25.mindspring.com>
Newsgroups: misc.int-property,misc.legal.computing,comp.os.linux.misc,comp.os.linux.misc,gnu.misc.discuss
Lines: 52

Roger Schlafly wrote:

> Karsten M. Self wrote in message <35C74BAC.547245AC@ix.netcom.com>...
> >Information at http://opensource.oreilly.com/townmeet.html
> >
> >Pamela Samuelson is a highly respected leading, if controversial,
> >intellectual property (IP) scholar who has criticized existing copyright
> >and patent protection of software. She argues for sui generis --
>
> If you attend, you should realize that her views are way out on
> the fringe, legally. She has written articles against all patent
> protection of software, and against all copyright protection
> of object code.

My view is somewhat similar. Though I respect her and her accomplishments, I will also say that I have gotten into some very heated software copyright debates with her in particular in the cni-copyright list, where she seemed to drop out after "losing" (in my mind) rather badly to the other copyright profs in that list.

In any case, my special criticism though is directed towards her position on software patents. If you read at least the older versions of her Manifesto, I think that you can best describe her position as not having a clue as to what is patentable and what is not as far as software is concerned.

Of course, the Manifesto can best be seen as a sales tool for her version of sui generis software protection legislation. The result there is arguably a slanted view of both software copyright and patent law, slanted to sell the need for sui generis legislation.

But of course, I disagree with the Manifesto to a very great extent. I find the combination of copyright and patent protection more than sufficient to protect software adequately. Indeed, I would suggest that many in this forum consider patent protection too strong for software, instead of not strong enough, as argued by Prof. Samuelson.
--

The preceding was not a legal opinion, and is not my employer's.
Original portions Copyright 1998 Bruce E. Hayden,all rights reserved
My work may be copied in whole or part, with proper attribution,
as long as the copying is not for commercial gain.

Bruce E. Hayden bhayden@acm.org
Phoenix, Arizona bhayden@copatlaw.com
***Currently use the following: bhayden@uswest.net

Subject: Re: Pamela Samuelson panelist at Open Source town meeting 8/21/98 San Jose, CA
Date: Wed, 05 Aug 1998 18:03:44 +0000
From: "Karsten M. Self" <kmself@ix.netcom.com>
Message-ID: <35C89E80.1745356F@ix.netcom.com>
References: <35C80D32.C7F5795F@uswest.net>
Newsgroups: misc.int-property,misc.legal.computing,comp.os.linux.misc,comp.os.linux.misc,gnu.misc.discuss
Lines: 66

Bruce Hayden wrote:
>
> Roger Schlafly wrote:
>
> > Karsten M. Self wrote in message <35C74BAC.547245AC@ix.netcom.com>...
> > >Information at http://opensource.oreilly.com/townmeet.html
> > >
> > >Pamela Samuelson is a highly respected leading, if controversial,
> > >intellectual property (IP) scholar who has criticized existing copyright
> > >and patent protection of software. She argues for sui generis --
> >
> > If you attend, you should realize that her views are way out on
> > the fringe, legally. She has written articles against all patent
> > protection of software, and against all copyright protection
> > of object code.
>

> Of course, the Manifesto can best be seen as a sales

> tool for her version of sui generis software protection
> legislation. The result there is arguably a slanted view of
> both software copyright and patent law, slanted to sell
> the need for sui generis legislation.
>
> But of course, I disagree with the Manifesto to a very
> great extent. I find the combination of copyright and
> patent protection more than sufficient to protect
> software adequately. Indeed, I would suggest that
> many in this forum consider patent protection too strong
> for software, instead of not strong enough, as argued
> by Prof. Samuelson.

Even I will admit to being a bit mystified by the Manifesto. Having started with her earlier paper, "CONTU Revisited" (Duke Law Journal c. 1984), which challenged the legal arguments for covering software under copyright, I'd gathered she generally felt that SW IP was headed for overprotection, and needs its own form of IP to correct this. The Manifesto makes the opposite case, a decade later -- SW IP is underprotected and needs its own form of IP to correct this. Me, I'm just sitting in the middle, confused.

I have to agree with the logical consistancy of many of the arguments made in "CONTU". OTOH, copyright seems to serve the needs of Open Source Software (OSS) very well, as for example in the GPL and other OSS licenses, so I won't complain too loudly. I am beginning to convince myself that OSS will prevail over patent by similar mechanims -- working within the existing law to protect itself from proprietary appropriation *and* patent challenge. Hopefully this will make both the lawyers and the programmers breath easier.

The Manifesto, interestingly, comments on the quantity and quality of "freeware" available in the SW world, though it doesn't make a major point of the issue.

> Bruce E. Hayden bhayden@uswest.net

--
Karsten M. Self (kmself@ix.netcom.com)

What part of "gestalt" don't you understand?
Welchen Teil von "gestalt" verstehen Sie nicht?

web: http://www.netcom.com/~kmself
SAS/Linux: http://www.netcom.com/~kmself/SAS/SAS4Linux.html

10:51am up 53 days, 8:20, 3 users, load average: 1.05, 1.23, 1.14

Subject: Re: Pamela Samuelson panelist at Open Source town meeting 8/21/98 San Jose, CA
Date: Wed, 5 Aug 1998 13:48:11 -0700
From: "Roger Schlafly" <nospam.schlafly@cruzio.com>
Message-ID: <6qageqvtk1@samsara0.mindspring.com>
References: <35C89E80.1745356F@ix.netcom.com>
Newsgroups: misc.int-property,misc.legal.computing,comp.os.linux.misc,comp.os.linux.misc,gnu.misc.discuss
Lines: 21

Karsten M. Self wrote in message <35C89E80.1745356F@ix.netcom.com>...
>Even I will admit to being a bit mystified by the Manifesto. Having
>started with her earlier paper, "CONTU Revisited" (Duke Law Journal c.
>1984), which challenged the legal arguments for covering software under
>copyright, I'd gathered she generally felt that SW IP was headed for
>overprotection, and needs its own form of IP to correct this. The
>Manifesto makes the opposite case, a decade later -- SW IP is
>underprotected and needs its own form of IP to correct this. Me, I'm
>just sitting in the middle, confused.

The argument for sui generis seems to depend on the claim that software is qualitatively different from other technologies. And yes it is, in some ways. But if this justifies sui gen, then it seems much easier to make the case that other technologies are different, and should have sui gen. Eg, genetic engineering, other biotech, integrated circuits, drugs, etc. Are there also law professors making a career out of arguing for sui generis protection of gene sequences?

Subject: Re: Pamela Samuelson panelist at Open Source town meeting 8/21/98 San Jose, CA
Date: Thu, 06 Aug 1998 06:02:23 +0000
From: "Karsten M. Self" <kmself@ix.netcom.com>
Message-ID: <35C946EF.975FB3D2@ix.netcom.com>
References: <6qageqvtk1@samsara0.mindspring.com>
Newsgroups: misc.int-property,misc.legal.computing,comp.os.linux.misc,comp.os.linux.misc,gnu.misc.discuss
Lines: 47

Roger Schlafly wrote:
>

> Karsten M. Self wrote in message <35C89E80.1745356F@ix.netcom.com>...
> >Even I will admit to being a bit mystified by the Manifesto. Having
> >started with her earlier paper, "CONTU Revisited" (Duke Law Journal c.
> >1984), which challenged the legal arguments for covering software under
> >copyright, I'd gathered she generally felt that SW IP was headed for
> >overprotection, and needs its own form of IP to correct this. The
> >Manifesto makes the opposite case, a decade later -- SW IP is
> >underprotected and needs its own form of IP to correct this. Me, I'm
> >just sitting in the middle, confused.
>
> The argument for sui generis seems to depend on the claim that
> software is qualitatively different from other technologies. And yes
> it is, in some ways. But if this justifies sui gen, then it seems much
> easier to make the case that other technologies are different,
> and should have sui gen. Eg, genetic engineering, other biotech,
> integrated circuits, drugs, etc. Are there also law professors making
> a career out of arguing for sui generis protection of gene sequences?

IC *got* sui generis protection with the SCPA (chip mask copyrighting statute, 17 U.S.C. 409). Most scholars, Samuelson included, argue that the was neither a "good" extension of copyright protection, nor necessary for protection of IC technology. I can't dig up a cite, but I believe she's written that the SCPA never resulted in a copyright infringement case.

Note also that SCPA protection is for two years for unregistered works, ten for registered. This is far shorter than the life + 50 years or 75 years of protection granted authors and corporations respectively under ordinary copyright.

I'm much less familiar with patent, but I believe biologicals have their own patent type. I'm not sure if this applies to bred plant species only or to biotechnology "inventions" in general.

--
Karsten M. Self (kmself@ix.netcom.com)

What part of "gestalt" don't you understand?
Welchen Teil von "gestalt" verstehen Sie nicht?

web: http://www.netcom.com/~kmself
SAS/Linux: http://www.netcom.com/~kmself/SAS/SAS4Linux.html

10:51pm up 53 days, 20:20, 2 users, load average: 1.06, 1.08, 1.03

Subject: Re: Pamela Samuelson panelist at Open Source town meeting 8/21/98 San Jose, CA
Date: Wed, 05 Aug 1998 23:52:29 -0700
From: Bruce Hayden <bhayden@ieee.org>
Message-ID: <35C952AD.F84D230A@ieee.org>
References: <6qageqvtk1@samsara0.mindspring.com>
Newsgroups: misc.int-property,misc.legal.computing,comp.os.linux.misc,comp.os.linux.misc,gnu.misc.discuss
Lines: 29

Roger Schlafly wrote:

> The argument for sui generis seems to depend on the claim that
> software is qualitatively different from other technologies. And yes
> it is, in some ways. But if this justifies sui gen, then it seems much
> easier to make the case that other technologies are different,
> and should have sui gen. Eg, genetic engineering, other biotech,
> integrated circuits, drugs, etc. Are there also law professors making
> a career out of arguing for sui generis protection of gene sequences?

And of course the example used for sui generis is semiconductor mask protection. It was obsolete almost from the date passed.
Yes, there is sui generis protection available for integrated circuits.
But it is today almost useless, if for no other reason that it is nearly impossible today to reconstruct a mask from an IC, esp. at the higher densities. Partly, this is because the etching wavelength is the same order of magnitude as the feature size.
--

The preceding was not a legal opinion, and is not my employer's.
Original portions Copyright 1998 Bruce E. Hayden,all rights reserved
My work may be copied in whole or part, with proper attribution,
as long as the copying is not for commercial gain.

Bruce E. Hayden bhayden@acm.org
Phoenix, Arizona bhayden@copatlaw.com
***Currently use the following: bhayden@uswest.net

Subject: Re: Pamela Samuelson panelist at Open Source town meeting 8/21/98 San Jose, CA
Date: Fri, 07 Aug 1998 18:45:18 +0000
From: "Karsten M. Self" <kmself@ix.netcom.com>
Message-ID: <35CB4B3E.36C1CF57@ix.netcom.com>
References: <6qfg66smo1@camel29.mindspring.com>

<6qf6ancgv1@csnews.cs.colorado.edu>
<6qageqvtk1@samsara0.mindspring.com>
Newsgroups: misc.int-property,misc.legal.computing,gnu.misc.discuss
Lines: 77

Roger Schlafly wrote:

> The idea is that when copyright and patent laws were conceived eons
> ago, no one could have anticipated changes in technology and
> society that have revolutionized the nature of the products being
> protected. Copyright protection is pretty much the same whether
> the work is a novel, painting, movie, or computer program.
>
> P. Samuelson takes the position that software is radically different,
> and requires special copyrights. She really wants much weaker
> software copyrights because she is sympathetic to the "free
> software" view of the world in which people who sell object code
> only are evil.

That's not quite my read.

- Samuelson argues that SW is radically different from other copyrighted works.

- She argues both that SW doesn't fit well within the copyright model, and that copyright offers little or no effective IP protection to SW.

SW doesn't fit copyright because copyright is designed to control publication while disseminating knowledge. SW uses copyright to control appropriation while disseminating product. Fair Use exceptions have emerged as the vehicle under which reverse engineering can occur, but it's still a bit messy, in my book.

IP provides no effective IP protection to SW because SW caries its IP "on its face" -- once reverse engineering has occured, the whole idea is presented, and the idea is not protected by copyright, only the expression of the idea.

- She argues that patent doesn't work well either for a number of reasons, including that it's too restrictive (17 years is for f------ ever in SW), too expensive, too slow (2-3 years for approval), and too hard to approve and ensure against infringement based on prior art and existing patents. Search mechanisms for both are apparently atrocious.

- Her latest work on the subject, "A Manifesto Concerning the Legal Protection of Computer Programs", 94 Columbia Law Review p 2308 (1994), argues that SW IP goes through periods of both over and under protection, based on copyright and patent, and that it really wants its own protection. The "Manifesto" reads to me as much more strong IP than her earlier "CONTU Revisited" (Duke L.J. 1984) which reviewed the original SW copyright act and history.

Though Samuelson has mentioned freeware and shareware (see Manifesto, p 2377), I haven't seen an unambiguous statement concerning her feelings on the subject. Her appearance on the OSS panel is very interesting for this reason.

AFAIK, much of her recent work has concerned WIPO, the various White and Green papers, and 2B. I suspect these will be the substance of her remarks at the Open Source Open House, though I've suggested to her that she address the issues of copyright and patent as well. Given the entire forum is 90 minutes and there are some half-dozen panelists, I doubt she'll have time for in-depth coverage of one topic, let alone three. The Q&A should be interesting though.

--
Karsten M. Self (kmself@ix.netcom.com)

What part of "gestalt" don't you understand?
Welchen Teil von "gestalt" verstehen Sie nicht?

web: http://www.netcom.com/~kmself
SAS/Linux: http://www.netcom.com/~kmself/SAS/SAS4Linux.html

11:31am up 55 days, 9:00, 2 users, load average: 1.17, 1.24, 1.10

Subject: Re: Pamela Samuelson panelist at Open Source town meeting 8/21/98 San Jose, CA
Date: Fri, 7 Aug 1998 12:44:00 -0700
From: "Roger Schlafly" <nospam.schlafly@cruzio.com>
Message-ID: <6qfleerni1@camel29.mindspring.com>
References: <35CB4B3E.36C1CF57@ix.netcom.com>
Newsgroups: misc.int-property,misc.legal.computing,gnu.misc.discuss
Lines: 47

Karsten M. Self wrote in message <35CB4B3E.36C1CF57@ix.netcom.com>...
> - Samuelson argues that SW is radically different from other
>copyrighted works.
>
> - She argues both that SW doesn't fit well within the copyright model,
>and that copyright offers little or no effective IP protection to SW.
>
>SW doesn't fit copyright because copyright is designed to control
>publication while disseminating knowledge. SW uses copyright to control
>appropriation while disseminating product. Fair Use exceptions have
>emerged as the vehicle under which reverse engineering can occur, but
>it's still a bit messy, in my book.

And do movies, music records, and artwork fit copyright?
This distinction seems bizarre to me.

>IP provides no effective IP protection to SW because SW caries its IP
>"on its face" -- once reverse engineering has occured, the whole idea is
>presented, and the idea is not protected by copyright, only the
>expression of the idea.

Now this is really absurd. How does she think Microsoft and
Oracle make their money?

> - She argues that patent doesn't work well either for a number of
>reasons, including that it's too restrictive (17 years is for f------
>ever in SW), too expensive, too slow (2-3 years for approval), and too
>hard to approve and ensure against infringement based on prior art and
>existing patents. Search mechanisms for both are apparently atrocious.

A lot of other fields have similar complaints against patents. Nothing
unique about SW here.

> - Her latest work on the subject, "A Manifesto Concerning the Legal
>Protection of Computer Programs", 94 Columbia Law Review p 2308 (1994),
>argues that SW IP goes through periods of both over and under
>protection, based on copyright and patent, and that it really wants its
>own protection. The "Manifesto" reads to me as much more strong IP than
>her earlier "CONTU Revisited" (Duke L.J. 1984) which reviewed the
>original SW copyright act and history.

Sounds like it would be more accurate to say that it is her own views
which have oscillated.

Subject: Re: Pamela Samuelson panelist at Open Source town meeting 8/21/98 San Jose,
CA
Date: Sat, 8 Aug 1998 02:04:32 -0700
From: "Roger Schlafly" <nospam.schlafly@cruzio.com>
Message-ID: <6qh4bb$1vs$1@camel29.mindspring.com>
References: <35CBF4CC.18894597@ix.netcom.com>
  <6qfleerni1@camel29.mindspring.com>
Newsgroups: misc.int-property,misc.legal.computing,gnu.misc.discuss
Lines: 50

Karsten M. Self wrote in message <35CBF4CC.18894597@ix.netcom.com>...
>None of these are utilitarian works, Samuelson's main argument. All
>carry, or reveal immediately when played, viewed, watched, whatever,
>their content. Publishers of records, producers of films or sculptors
>of artwork don't seek to keep the buyer of a copy of the work from
>examining it in complete detail. Software -- object code -- by
>contrast, requires detailed examination and mechanical processing to
>divulge its meaning.

A sculptor doesn't necessarily reveal his techniques, nor is it obvious
from the sculpture. Likewise for many other works. I find this
distinction unpersuasive.

>> >IP provides no effective IP protection to SW because SW caries its IP
>> >"on its face" -- once reverse engineering has occured, the whole idea is
>> >presented, and the idea is not protected by copyright, only the
>> >expression of the idea.
>>
>> Now this is really absurd. How does she think Microsoft and
>> Oracle make their money?
>
>Copyright protects against direct copying of software -- I can burn a
>CD-ROM and sell it, but not legally. Copyright does not prevent me from
>figuring out how a program works and utilizing the ideas in my own
>creation. This takes time.

You are not answering the question. If PS says that SW has no
effective IP protection, then she is an idiot. The IP protection is
strong enough for Microsoft and Oracle to make billions of dollars.

>I said there was? No, I said that Samueson doesn't think patent does a
>good job with software. Just because my grass is green doesn't mean
>yours is or isn't.

She is not a patent attorney, and is in over her head on this subject.

It is useless to complain about SW patents when other patents have the same problems.

> Software currently has far less protection under
>court interpretation of copyright than it did in 1984. See Lotus v.
>Borland, Sega v. Accolade, Atari v. Nintendo.

It sounds like you have been listening to Greenberg. You and he are wrong about this. Those decisions did not reverse existing law. SW copyright protection is every bit as strong now as it was in 1984.

Subject: Re: Pamela Samuelson panelist at Open Source town meeting 8/21/98 San Jose, CA
Date: Sat, 08 Aug 1998 19:45:28 +0000
From: "Karsten M. Self" <kmself@ix.netcom.com>
Message-ID: <35CCAAD8.22B95A6A@ix.netcom.com>
References: <6qh4bb$1vs$1@camel29.mindspring.com>
Newsgroups: misc.int-property,misc.legal.computing,gnu.misc.discuss
Lines: 130

Roger Schlafly wrote:
>
> Karsten M. Self wrote in message <35CBF4CC.18894597@ix.netcom.com>...
> >None of these are utilitarian works, Samuelson's main argument. All
> >carry, or reveal immediately when played, viewed, watched, whatever,
> >their content. Publishers of records, producers of films or sculptors
> >of artwork don't seek to keep the buyer of a copy of the work from
> >examining it in complete detail. Software -- object code -- by
> >contrast, requires detailed examination and mechanical processing to
> >divulge its meaning.
>
> A sculptor doesn't necessarily reveal his techniques, nor is it obvious
> from the sculpture. Likewise for many other works. I find this
> distinction unpersuasive.

Nor does a book reveal the bookbinding techniques. The protection granted by copyright for a book, a sculture, a musical work, or an architectural design is for the *expression* of the work, not the construction of the work. A sculptor's copyright does not protect his techniques. An author's copyright does not protect her research methods. An architect's copyright protects neither her drafting methods nor construction techniques. Red herring.

As I see it, most software produces a number of artifacts (I'm speaking of produced or derived objects, outside the scope of copyright):
- Design documentation (schemas, notes)
- Source code
- Object code
- Runtime instance -- interaction of OC with dynamic libraries, other environmental factors, runtime images in RAM, cache, VM, etc.
- Runtime actions, data, text, displays, screens, audio, visual, or other outputs (will vary by type of program, some or all may not be applicable).
- Latent outputs -- permanent data output, hardcopy, email, etc.

The copyright dilemma (I'm not saying flaw, I'm not saying fatal error, I'm saying dilemma) is this:

- It is the runtime actions of a program, and the latent outputs, which are the valuable attributes of a program. All else is overhead.

- It is the source code which provides the ultimate "design document" of the program. This is pretty arguably a "literary work". However it is the object code which directly enables a runtime instance. The argument that object code is itself a literary work is weakened by the near total impossiblity to gain meaningful information by direct reading of the object code.

Returning to the argument: So if "source code" is techniques, and "object code" is a "derived work" of source, if both are covered under copyright, and if object code is published and distributed, why should copyright protection cover anything other than the *expression* of technique under software. The law has held that it does not (Sega v. Accolade), and that reverse engineering for purposes of identifying methods and mechanisms.

To restate: software carries its technological content near its face. This technology is readily discoverable. Copyright cannot legally provide a bar to discovering this content by 17 U.S.C. 102(b) (utilitarian exemption) and 107 (fair use). Much of the legal history is the story of copyright holders seeking strong IP protection against alleged infringers seeking week IP protection under copyright. The current status is that gross appropriation is considered infringing, but reverse engineering is allowed.

> >> >IP provides no effective IP protection to SW because SW caries its IP
> >> >"on its face" -- once reverse engineering has occured, the whole idea is
> >> >presented, and the idea is not protected by copyright, only the
> >> >expression of the idea.
> >>
> >> Now this is really absurd. How does she think Microsoft and
> >> Oracle make their money?
> >
> >Copyright protects against direct copying of software -- I can burn a
> >CD-ROM and sell it, but not legally. Copyright does not prevent me from
> >figuring out how a program works and utilizing the ideas in my own
> >creation. This takes time.
>
> You are not answering the question. If PS says that SW has no
> effective IP protection, then she is an idiot. The IP protection is
> strong enough for Microsoft and Oracle to make billions of dollars.

Let me change history. I meant to write "Copyright provides no
effective IP protection...CRYPHTML.HTM" This is true in law and in fact.

The fact that the industry thrives despite this lack might be a good
argument that the weak protections curretnly provided are sufficient.

> >I said there was? No, I said that Samueson doesn't think patent does a
> >good job with software. Just because my grass is green doesn't mean
> >yours is or isn't.
>
> She is not a patent attorney, and is in over her head on this subject.
> It is useless to complain about SW patents when other patents
> have the same problems.

The fact is that she thinks patent doesn't fit SW well. You can agree
or disagree. You can say other areas of IP have similar problems. If
the problem is alledged for SW, you can provide confirming or counter
evidence to support her views, or evidence that she's unqualified to
hold an informed opinion. I was simply correcting your gross
misstatements of her views.

> > Software currently has far less protection under
> >court interpretation of copyright than it did in 1984. See Lotus v.
> >Borland, Sega v. Accolade, Atari v. Nintendo.
>
> It sounds like you have been listening to Greenberg. You and he
> are wrong about this. Those decisions did not reverse existing
> law. SW copyright protection is every bit as strong now as it
> was in 1984.

Compare and contrast Lotus v. Borland and Lotus v. Paperback.

--
Karsten M. Self (kmself@ix.netcom.com)

What part of "gestalt" don't you understand?
Welchen Teil von "gestalt" verstehen Sie nicht?

web: http://www.netcom.com/~kmself
SAS/Linux: http://www.netcom.com/~kmself/SAS/SAS4Linux.html

12:11pm up 56 days, 9:40, 2 users, load average: 1.19, 1.18, 1.11

Subject: Re: Pamela Samuelson panelist at Open Source town meeting 8/21/98 San Jose,
CA
Date: Sun, 09 Aug 1998 01:59:22 +0000
From: "Karsten M. Self" <kmself@ix.netcom.com>
Message-ID: <35CD027A.4240A826@ix.netcom.com>
References: <6qijtg$vm0$1@camel25.mindspring.com>
<35CCAAD8.22B95A6A@ix.netcom.com>
Newsgroups: misc.int-property,misc.legal.computing,gnu.misc.discuss
Lines: 79

This is my final post in this thread.

Roger Schlafly wrote:
>
> Karsten M. Self wrote in message <35CCAAD8.22B95A6A@ix.netcom.com>...
> >Nor does a book reveal the bookbinding techniques. The protection
> >granted by copyright for a book, a sculture, a musical work, or an
> >architectural design is for the *expression* of the work, not the
> >construction of the work. A sculptor's copyright does not protect his
> >techniques. An author's copyright does not protect her research
> >methods. An architect's copyright protects neither her drafting methods
> >nor construction techniques. Red herring.
>
> Right. Same for software. What's your beef?

If you want to review my previous arguments, which is where the beef is,
then re-read the posts. Don't ask stupid questions.

> >Let me change history. I meant to write "Copyright provides no
> >effective IP protection...CRYPHTML.HTM" This is true in law and in fact.

>
> I've tried to be patient with you, but anyone who would say that
> is just idiot. Have you ever heard of Bill Gates? He is America's
> richest man. He got there using the effective IP protection that
> he got from copyright law.

Which would argue that existing IP protections, weak or strong, are
sufficient to foster creation of an industry in which one man may amass
a wealth of $50 billion.

Bill got to where he is by several means. Marketing is a word I hear
frequently. The IP-is-the-source-of-Bill's-wealth school has few
adherants.

> >> > Software currently has far less protection under
> >> >court interpretation of copyright than it did in 1984. See Lotus v.
> >> >Borland, Sega v. Accolade, Atari v. Nintendo.
> >>
> >> It sounds like you have been listening to Greenberg. You and he
> >> are wrong about this. Those decisions did not reverse existing
> >> law. SW copyright protection is every bit as strong now as it
> >> was in 1984.
> >
> >Compare and contrast Lotus v. Borland and Lotus v. Paperback.
>
> Lotus won both in district court with a judge who tried to stake out
> new law with his opinion. His novel interpretation of copyright
> was never accepted and reversed on appeal. The main
> difference is that Paperback settled the case before appealing.

Lotus won in Paperback (the other side capitulated). Mid-late 1980's.
Lotus lost in Borland, early 1990's.

Claim was roughly the same -- different source code gave rise to similar
programs (Paperback was a near total clone, Borland had a menu/function
compatibility mode).

Samuelson cites the Whelan v. Jaslow case as the high water of strong
copyright SW IP protection.

Lotus v. Borland, Sega v. Accolade, and CA v. Altai have significantly
weakened the level of protection afforded by copyright. Borland said
that program elements may be duplicated without infringement, Accolade
that reverse engineering is allowable under Fair Use, and Altai that AFC
was the means for seperating copyrightable expression from
non-copyrightable ideas in a program.

--
Karsten M. Self (kmself@ix.netcom.com)

    What part of "gestalt" don't you understand?
    Welchen Teil von "gestalt" verstehen Sie nicht?

web: http://www.netcom.com/~kmself
SAS/Linux: http://www.netcom.com/~kmself/SAS/SAS4Linux.html

  6:31pm up 56 days, 16:00, 3 users, load average: 2.06, 2.02, 2.00

Subject: Re: Pamela Samuelson panelist at Open Source town meeting 8/21/98 San Jose,
CA
Date: Sun, 09 Aug 1998 14:24:01 -0400
From: werdna@nonspam-gate.net (Andrew C. Greenberg)
Message-ID: <werdna-0908981424010001@tstpa1-61.gate.net>
References: <35CD027A.4240A826@ix.netcom.com>
Newsgroups: misc.int-property,misc.legal.computing,gnu.misc.discuss
Lines: 55

In article <35CD027A.4240A826@ix.netcom.com>, kmself@ix.netcom.com wrote:

> Roger Schlafly wrote:

> > >Let me change history. I meant to write "Copyright provides no
> > >effective IP protection...CRYPHTML.HTM" This is true in law and in fact.
> >
> > I've tried to be patient with you, but anyone who would say that
> > is just idiot. Have you ever heard of Bill Gates? He is America's
> > richest man. He got there using the effective IP protection that
> > he got from copyright law.

Indeed, Copyright protection of the literal expression (meaning
byte-for-byte copying) has been unchanged through the ages -- it is
difficult to pirate Microsoft's works of authorship without committing
Copyright infringement, and that hasn't changed substantially over the
years.

However, it is plain beyond cavil to any serious student of this area of
law that the scope of non-literal infringement has diminished dramatically
over the past twenty years. Indeed, it has diminished in some
jurisdictions to practically nothing. (See the last three 11th Circuit
opinions on the subject, adopting a "virtual identity" test for certain
types of non-literal software infringement).

Roger's patience is irrelevant. He is a foolish, boorish man who doesn't really know what to do but exercise gainsay to forward his particular ideologies as to what the law "should" be. Karsten accurately represents what most writers consider to be the state of the law, for the reasons and authority set forth in Karsten's postings (not responded to on the merits by Roger) and Roger's gainsay, however often he might repeat it, cannot change that fact.

I have noted in Karsten's postings (I no longer read Roger's -- he is one of a select few whose signal-to-noise ratio is so low as to earn a spot on my bozo list), it appears Roger made a number of truthful and accurate, at times, salient points throughout this colloquy. This wasn't one of them. Readers looking for a credible gaff-o-meter can generally know when Roger is trolling by the fact that he (1) begins ad hominems and repeates them; and (2) stops actually arguing on the merits.

Readers should simply ignore Mr. Schlafly's remarks in this regard. He simply doesn't have a clue what he is talking about. The contrary position is well-recognized by those who routinely deal in this arena or have educated themselves as to what actually is the law. A recent National Law Journal article discussing the ever-diminishing importance of copyrights as an arrow in the software IP quiver (for non-literal infringement) can be found at:

http://www.ljextra.com/practice/computer/0420softpat.html

--
just another view,
Andy Greenberg
(remove "nonspam-" to get my real mailing address)

Subject: Re: Pamela Samuelson panelist at Open Source town meeting 8/21/98 San Jose, CA
Date: Sun, 09 Aug 1998 14:05:21 -0400
From: werdna@nonspam-gate.net (Andrew C. Greenberg)
Message-ID: <werdna-0908981405210001@tstpa1-61.gate.net>
References: <35CCAAD8.22B95A6A@ix.netcom.com>
Newsgroups: misc.int-property,misc.legal.computing,gnu.misc.discuss
Lines: 31

In article <35CCAAD8.22B95A6A@ix.netcom.com>, kmself@ix.netcom.com wrote:

> Roger Schlafly wrote:

> > > Software currently has far less protection under
> > >court interpretation of copyright than it did in 1984. See Lotus v.
> > >Borland, Sega v. Accolade, Atari v. Nintendo.
> >
> > It sounds like you have been listening to Greenberg. You and he
> > are wrong about this. Those decisions did not reverse existing
> > law. SW copyright protection is every bit as strong now as it
> > was in 1984.
>
> Compare and contrast Lotus v. Borland and Lotus v. Paperback.

Gentle readers will notice the difference between the reasonable (albeit misguided but anxious to study and learn IMHO :-)) poster, Mr. Self, who has produced now at least five distinct authorities, and the trolling demagogue, whose sole argument comprises an ad hominem attack and naked assertion of correctness.

Roger has no education or credentials in copyright law to speak of, and by his remarks and comments in this newsgroup throughout the years, one can reasonably conclude his understanding of these areas of law is nominal to weak. In short, Roger can be relied to come up with whatever argument suits his current political views, regardless of what the law may be.

--
just another view,
Andy Greenberg
(remove "nonspam-" to get my real mailing address)

Subject: Re: Pamela Samuelson panelist at Open Source town meeting 8/21/98 San Jose, CA
Date: Sat, 08 Aug 1998 08:56:49 -0400
From: werdna@nonspam-gate.net (Andrew C. Greenberg)
Message-ID: <werdna-0808980856490001@tstpa1-18.gate.net>
References: <35CBF4CC.18894597@ix.netcom.com>
<6qfleerni1@camel29.mindspring.com>
Newsgroups: misc.int-property,misc.legal.computing,gnu.misc.discuss
Lines: 166

In article <35CBF4CC.18894597@ix.netcom.com>, kmself@ix.netcom.com wrote:

> Roger Schlafly wrote:
> >
> > Karsten M. Self wrote in message <35CB4B3E.36C1CF57@ix.netcom.com>...
>
>
> You'd made a statement on Samuelson's views. I disagreed, gave my take
> on her position. Her position and the reasons for it, she'd have to
> answer for.

>
>
> > > - Samuelson argues that SW is radically different from other
> > >copyrighted works.
> > >
> > > - She argues both that SW doesn't fit well within the copyright model,
> > >and that copyright offers little or no effective IP protection to SW.
> > >
> > >SW doesn't fit copyright because copyright is designed to control
> > >publication while disseminating knowledge. SW uses copyright to control
> > >appropriation while disseminating product. Fair Use exceptions have
> > >emerged as the vehicle under which reverse engineering can occur, but
> > >it's still a bit messy, in my book.
> >
> > And do movies, music records, and artwork fit copyright?
> > This distinction seems bizarre to me.
>
> None of these are utilitarian works, Samuelson's main argument.

Define "utilitarian." Does it extend to factual accounts and
non-fiction? Are my computer games "utilitarian?" Here is where Paula's
arguments begin to fall on their own weight. [Be prepared for a
discussion of architectural works.]

> All
> carry, or reveal immediately when played, viewed, watched, whatever,
> their content. Publishers of records, producers of films or sculptors
> of artwork don't seek to keep the buyer of a copy of the work from
> examining it in complete detail. Software -- object code -- by
> contrast, requires detailed examination and mechanical processing to
> divulge its meaning.

That's, at most, a distinction without a difference. Many works are
perceived not by the expression directly, but by the "execution" of the
expression. Records and tape recordings, written music, written dance
recordations, scripts, screenplays. Noone purchasing and using software
CARES about the particular expressions of the mechanical object
code/binary format for recording the record/use of notation to relate
music or dance or theatrical and stage dirctions -- what they enjoy is the
execution of those things.

> > >IP provides no effective IP protection to SW because SW caries its IP
> > >"on its face" -- once reverse engineering has occured, the whole idea is
> > >presented, and the idea is not protected by copyright, only the
> > >expression of the idea.
> >
> > Now this is really absurd. How does she think Microsoft and
> > Oracle make their money?
>
> Copyright protects against direct copying of software -- I can burn a
> CD-ROM and sell it, but not legally. Copyright does not prevent me from
> figuring out how a program works and utilizing the ideas in my own
> creation. This takes time.

This is nonresponsive to the poster's point, which is that IP,
particularly copyright, provides very effective protection indeed for the
copyright owner's expression. Otherwise, Microsoft would not be as
financially well-endowed as many nations.

> > > - She argues that patent doesn't work well either for a number of
> > >reasons, including that it's too restrictive (17 years is for f------
> > >ever in SW), too expensive, too slow (2-3 years for approval), and too
> > >hard to approve and ensure against infringement based on prior art and
> > >existing patents. Search mechanisms for both are apparently atrocious.
> >
> > A lot of other fields have similar complaints against patents. Nothing
> > unique about SW here.
>
> I said there was? No, I said that Samueson doesn't think patent does a
> good job with software. Just because my grass is green doesn't mean
> yours is or isn't.

Right, but its a valid point anyhow. The speed of the system and its
length of term is the same for every art group -- yet patents work great
with the others. Software needs first to be distinguished from those
other groups before the argument can be meaningful. Again, we need to
consider whether the argument being made is: (1) patents are bad; (2)
software patents are bad; or (3) bad software patents are bad. If this is
a (1) argument, its fairly laughable because the patent system works just
great and demonstrably so. If this is a (2) argument, a huge jump was
taken without explanation -- why are these issues unique o software?

> > > - Her latest work on the subject, "A Manifesto Concerning the Legal
> > >Protection of Computer Programs", 94 Columbia Law Review p 2308 (1994),
> > >argues that SW IP goes through periods of both over and under
> > >protection, based on copyright and patent, and that it really wants its
> > >own protection. The "Manifesto" reads to me as much more strong IP than
> > >her earlier "CONTU Revisited" (Duke L.J. 1984) which reviewed the
> > >original SW copyright act and history.
> >
> > Sounds like it would be more accurate to say that it is her own views
> > which have oscillated.
>
> Ask her. I'm a bit puzzled. Her second line is consistant with the
> first, in context. Software currently has far less protection under

> court interpretation of copyright than it did in 1984. See Lotus v.
> Borland, Sega v. Accolade, Atari v. Nintendo.

If what she is saying is that the scope of IP protection takes some time to adjust for emerging technology, and this means that the emerging technology requires sui generis protection, she's got a VERY weak argument. This oscillation of protection is true FOR EVERY NEW ART AREA AS A MATTER OF COURSE.

IP Laws are designed to be broad and flexible -- to adapt to changes in the embodiments of their respective subject matter. And yes, this takes time. But the protection didn't realy change -- just the "betting odds" on how it would shake out. "Look and Feel" was never the law, just an optimistic sense as to where some overreadings of dicta in some New England opinions might lead. AFC was proposed long ago, but the basic principles of law have not changed.

As the cases developed, we knew more clearly where the law was, and what were safe harbors not worth litigating in the future, but so what? And please note, the changes were not changes adapting to the unique nature of software, in my view, but rather changes RECOGNIZING HOW SOFTWARE WAS NOT DIFFERENT. AFC is simply an application of well-understood case law dealing with movies and plays to software, not some change in the law to accomodate the differences between software and all other works. I believe the present cases are actually settling into the idea that software really ISN'T all that different from other subject matter (for example as the "look and feel" approach would have required; nothign else is protected that way), not that it is fundamentally different.

Moreover, why change NOW? What makes us think that we have a comprehension today of this area of law that justifies making a change -- when market conditions and technologies are just as likely to be different at tht time the bill is passed (just as happened in seminconductor masks)?

Adding a new body of law simply makes it more expensive to get relief, and will require yet another two decades before there is any certainty as to what the statutes would mean. Having warred out the meanings of new statutory langauge in the face of no decided case law, trust me, new law, however clearly written will cost MILLIONS in fees before it is "sorted."

I believe that the idea of sui generis protection is by itself very dangerous, even if the law is better than the status quo in theory. Without getting to the merits, what of the chaos and uncertainty that will ensue? Yes, this is a highly conservative position, but after studying Copyright Law since the 70's and seeing it evolve, I wouldn't want to give up the certainty we now have as to how to govern our conduct for anything.

Paula wants to make a change, but she doesn't really say why change is necessary. She observes the fit between patents, copyrights and software isn't perfect. True, but when is the "fit' perfect? The question is whether it is doing its job. I don't see the schrapnel falling that the critics claimed would happen -- none of it. I do, however, see a lot of good arising from software IP. It would take a compelling argument, not a whine about aesthetic objections or a laundry list of relatively insignificant difference, to lead me to think new legislation is called for or justified.

--
just another view,
Andy Greenberg
(remove "nonspam-" to get my real mailing address)

Subject: Re: Pamela Samuelson panelist at Open Source town meeting 8/21/98 San Jose, CA
Date: Sat, 08 Aug 1998 20:23:57 +0000
From: "Karsten M. Self" <kmself@ix.netcom.com>
Message-ID: <35CCB3DD.9F1EB9BE@ix.netcom.com>
References: <werdna-08089808564900001@tstpa1-18.gate.net>
Newsgroups: misc.int-property,misc.legal.computing,gnu.misc.discuss
Lines: 191

Andrew C. Greenberg wrote:

Let me start at the end:

> If what she is saying is that the scope of IP protection takes some time
> to adjust for emerging technology, and this means that the emerging
> technology requires sui generis protection, she's got a VERY weak
> argument. This oscillation of protection is true FOR EVERY NEW ART AREA
> AS A MATTER OF COURSE.

I'm in general agreement with this argument. I may differ slightly in degree, but I am not convinced that a new area of law must be carved out for software vis-a-vis copyright. If anything, the traditional area of copyright seems to be following software into the electronic age. The current legislation is largely aimed at this (mis-aimed, in my book, but...).

> I believe that the idea of sui generis protection is by itself very
> dangerous, even if the law is better than the status quo in theory.
> Without getting to the merits, what of the chaos and uncertainty that will
> ensue? Yes, this is a highly conservative position, but after studying

> Copyright Law since the 70's and seeing it evolve, I wouldn't want to give
> up the certainty we now have as to how to govern our conduct for anything.

Don't rock the boat? There are some merits, but I'd be loathe to
promote this as my primary objection to manifestly bad law. Say, civil
rights, women's rights reforms? I could see phase-in periods for new
law to allow for orderly transition, but to say "we can't do that, it
would be better to be there but it might hurt to get there" is weak.
Say a 5-10 year period of transition.

> Paula wants to make a change, but she doesn't really say why change is
> necessary. She observes the fit between patents, copyrights and software
> isn't perfect. True, but when is the "fit' perfect? The question is
> whether it is doing its job. I don't see the schrapnel falling that the
> critics claimed would happen -- none of it. I do, however, see a lot of
> good arising from software IP. It would take a compelling argument, not a
> whine about aesthetic objections or a laundry list of relatively
> insignificant difference, to lead me to think new legislation is called
> for or justified.

I still see some rough edges, particularly in the object code as
copyrighted debate, infringing RAM copies (effects much more than
software, see above on traditional areas of copyright). SW licensing
and EULAs have their warts. Patent takes too long to get, lasts too
long when got, isn't checked sufficiently when registerd, and is hard to
challenge and unregister. Industrial damage as a result? Not sure.

> In article <35CBF4CC.18894597@ix.netcom.com>, kmself@ix.netcom.com wrote:
>
> > Roger Schlafly wrote:
> > >
> > > Karsten M. Self wrote in message <35CB4B3E.36C1CF57@ix.netcom.com>...

> > > >SW doesn't fit copyright because copyright is designed to control
> > > >publication while disseminating knowledge. SW uses copyright to control
> > > >appropriation while disseminating product. Fair Use exceptions have
> > > >emerged as the vehicle under which reverse engineering can occur, but
> > > >it's still a bit messy, in my book.
> > >
> > > And do movies, music records, and artwork fit copyright?
> > > This distinction seems bizarre to me.
> >
> > None of these are utilitarian works, Samuelson's main argument.
>
> Define "utilitarian." Does it extend to factual accounts and
> non-fiction? Are my computer games "utilitarian?" Here is where Paula's
> arguments begin to fall on their own weight. [Be prepared for a
> discussion of architectural works.]

How about:

utilitarian: (3a) of, relating to, or aiming at utility
utility: (1) fitness for some purpose or worth to some end. (2)
Something useful or designed for use.
[Webster's Ninth Collegiate Dictionary]

A "useful article" is an article having an intrinsic utilitarian
function that is not merely to portray the appearance of the
article or to convey information. An article that is normally a
part of a useful article is considered a "useful article".

"Pictorial, graphic, and sculptural works" include two-dimensional
and three-dimensional works of fine, graphic, and applied art,
photographs, prints and art reproductions, maps, globes, charts,
diagrams, models, and technical drawings, including architectural
plans. Such works shall include works of artistic craftsmanship
insofar as their form but not their mechanical or utilitarian
                      ^^
aspects are concerned; the design of a useful article, as defined
in this section, shall be considered a pictorial, graphic, or
sculptural work only if, and only to the extent that, such design
incorporates pictorial, graphic, or sculptural features that can
be identified separately from, and are capable of existing
independently of, the utilitarian aspects of the article.
^^^

A "computer program" is a set of statements or instructions to be
used directly or indirectly in a computer in order to bring about
a certain result.

[17 U.S.C. 101]

The three paragraphs from 17 U.S.C. 101 have long bothered me. They
don't seem to fit together. Note that the pictoral, graphic, and
sculptural works definition excludes computer programs, independently
defined. It's the spirit, not the letter, I'm emphasizing.

I don't understand your question WRT factual or fictional accounts. If
you are talking about works about factual or fictional accounts, I would

say that there may be some utility in the information, but that utility is not protected by copyright -- merely the expression of the fictional/factual account. Characters, situations, and themes are copyrightable, under the courts.

I would deem your games utilitarian insofar as they are comprised of utilitarian interactions and modifications of technological artifacts -- computer software and hardware.

I'll counter: what is your prespective of the sale of a game? Is the customer buying the program which makes the game possible, or are they buying the instantiation created when the program is run? Which of these -- code or game -- would you rather have protected on the market?

> > All
> > carry, or reveal immediately when played, viewed, watched, whatever,
> > their content. Publishers of records, producers of films or sculptors
> > of artwork don't seek to keep the buyer of a copy of the work from
> > examining it in complete detail. Software -- object code -- by
> > contrast, requires detailed examination and mechanical processing to
> > divulge its meaning.
>
> That's, at most, a distinction without a difference. Many works are
> perceived not by the expression directly, but by the "execution" of the
> expression. Records and tape recordings, written music, written dance
> recordations, scripts, screenplays. Noone purchasing and using software
> CARES about the particular expressions of the mechanical object
> code/binary format for recording the record/use of notation to relate
> music or dance or theatrical and stage dirctions -- what they enjoy is the
> execution of those things.

....the hackers do.

I'm not buying the instantiation of the program, I'm buying the object code which makes the instantiation possible. See my response to Roger for more discussion.

Audio and visual recordings are different in that the information provided is not interpreted "to bring about a certain result" -- it's simply played back directly, with a very direct, one-to-one mapping between recorded data and revealed information.

Note again that this distinction is lost with more complex forms of recordings such as multimedia and interactive books. Again, I see non-SW copyright issues converging on the concerns of SW.

> This is nonresponsive to the poster's point, which is that IP,
> particularly copyright, provides very effective protection indeed for the
> copyright owner's expression. Otherwise, Microsoft would not be as
> financially well-endowed as many nations.

It was you yourself who has pointed out to me several times that copyright provides *no* protection of the ideas embodied in software.

I answered this in response to Roger. Suggest that IP is at least sufficiently strong if the industry is thriving as much as it is.

> Andy Greenberg

--
Karsten M. Self (kmself@ix.netcom.com)

What part of "gestalt" don't you understand?
Welchen Teil von "gestalt" verstehen Sie nicht?

web: http://www.netcom.com/~kmself
SAS/Linux: http://www.netcom.com/~kmself/SAS/SAS4Linux.html

12:41pm up 56 days, 10:10, 2 users, load average: 1.26, 1.28, 1.18

Subject: Re: Pamela Samuelson panelist at Open Source town meeting 8/21/98 San Jose, CA
Date: Sun, 09 Aug 1998 14:00:50 -0400
From: werdna@nonspam-gate.net (Andrew C. Greenberg)
Message-ID: <werdna-09089814005000001@tstpa1-61.gate.net>
References: <35cc6c9f.978749@news.ucla.edu>
<werdna-08089808564900001@tstpa1-18.gate.net>
Newsgroups: misc.int-property,misc.legal.computing
Lines: 33

In article <35cc6c9f.978749@news.ucla.edu>, bstock@ucla.edu (Bob Stock) wrote:

> This is true, but it doesn't go far enough. Samuelson's point is that
> that you can have execution of programs with different "expression,"
> but which both produce identical behavior. You can't do that with
> musical or textual symbols.

That's nonsense, of course. Different programs (modulo comments that do

not compile to code or mechanical substitutions of names) typically have different executions. Sometimes they make a difference, and sometimes they don't.

But it is the same with music, drama and dance. What is the difference between (Brief Pause) and (Beat) in a script? Between piano and pianissimo? Between writing a repeat using repeat symbols, or a D.S. with Coda, and rewriting a phrase out entirely? Between different phrasing notations, between different barrings? About 3,000,000 different digital representations of the same musical performance? About 3,000,000 different mixes of the same musical performance?

Different expressions can yield, at an appropriate level of abstractions, dramatically similar works, and different expressions, at an appropriate level of abstraction can yield "clearly" different works.

And besides, Bob, what's your point? Why does the fact that two different "expressions" yield an identical "behavior" (presumably defined at the appropriate level of abstraction) have to do with the propriety or scope of I.P. protection?

--
just another view,
Andy Greenberg
(remove "nonspam-" to get my real mailing address)

Subject: Re: Pamela Samuelson panelist at Open Source town meeting 8/21/98 San Jose, CA
Date: 5 Aug 1998 14:24:06 -0500
From: peltz@jaka.ece.uiuc.edu (Steve Peltz)
Message-ID: <6qabgm$1cp$1@jaka.ece.uiuc.edu>
References: <35C89C0D.479E1D47@ix.netcom.com>
<6q9nrc$jj2@bourbon.cs.umd.edu>
<6q8nc2$8n8$1@camel25.mindspring.com>
Newsgroups:
misc.int-property,misc.legal.computing,comp.os.linux.misc,gnu.misc.discuss
Lines: 56

In article <35C89C0D.479E1D47@ix.netcom.com>,
Karsten M. Self <kmself@ix.netcom.com> wrote:
>Name three SW patents, including patent number, which you feel are
>unreasonable. (XOR and Pantone have been discussed).

The Wang patent. The backing-store patent (I "invented" it myself shortly after finding out there WERE graphical windowing systems (that didn't use it, due to the high price of memory at the time) - so I feel I can honestly assert that it was obvious). Using a character ROM on a glass TTY.

Two I've been involved in were a DEC patent on "ring buffers" and a patent on a bingo game.

The DEC patent is 4,449,182, issued May 15, 1984, INTERFACE BETWEEN A PAIR OF PROCESSORS SUCH AS HOST AND PERIPHERAL-CONTROLLING PROCESSORS IN DATA PROCESSING SYSTEMS. I'll have to go find the bingo patent number (I just re-found the copy of the deposition; I'm pretty sure the patent number is in there).

As far as I know, both patents are still in force, although successful defenses have been raised against them (i.e. cases were settled out of court). Someone called me a few months ago about the bingo patent, so I'm pretty sure it is still going strong.

Both had relevant prior art presented; in the bingo case, one ruling by the judge was that since the patented game used distributed processing (the user terminal did the logic to determine whether a BINGO had been scored), and the prior art version was implemented on a single mainframe (even though, internally, the game was implemented as separate processes communicating through a "communications means", in essentially the exact same way as the patent described, regardless of whether that was through shared memory buffers or a network), and didn't automatically call a BINGO (intentionally - the whole point was to make the PLAYER notice it and press a key; at which point the program would verify if it was or not).

The DEC patent was pretty much blown out of the water. We were using the same method, in a publically used system, well before the patent. To top it off, we still had on-line discussions of the changes that were made and why, which matched pretty closely the advantages claimed in the patent. Yet the patent is still in force as far as I know (and I think DEC still claims they've never lost a patent case).

Regardless of prior art, I would claim that BOTH patents were "obvious" (given the current state of the art at the time), almost trivial in fact, and not worthy of patent protection AND were being abused (the DEC case was using it as a protection against being able to build compatible peripherals, as the OS used that method for communicating with external devices).

Actually, the bingo patent was OK on the face of it - it is only when it is interpreted as a software patent that it became a problem. It had a pretty clever method of using a physical game board with the layout of that board encoded into edge contacts to let the computer determine when a BINGO could be called. If it stuck with that aspect only, I wouldn't

have a problem with it.

Subject: Re: Pamela Samuelson panelist at Open Source town meeting 8/21/98 San Jose, CA
Date: Thu, 06 Aug 1998 22:17:02 -0400
From: werdna@nonspam-gate.net (Andrew C. Greenberg)
Message-ID: <werdna-0608982217020001@tstpa1-110.gate.net>
References: <6qci3qmmh1@nnrp1.dejanews.com>
<35C9576B.7CC3675A@ieee.org>
Newsgroups:
misc.int-property,misc.legal.computing,comp.os.linux.misc,comp.os.linux.misc
Lines: 27

In article <6qci3qmmh1@nnrp1.dejanews.com>, johnny_squire@hotmail.com wrote:

> 1. A patent monopoly is given in exchange for a valuable disclosure to the
> public.
> 2. For a disclosure to be valuable, the public must learn from it.
> 3. The public learns nothing from software patent disclosures (i.e. name three
> programmers who ever learned anything by reading it in a patent).
> Therefore a patent monopoly is inappropriate in exchange for software patent
> disclosures.

Big leap there.

If the disclosure is directed to an enabling disclosure of a new, useful and unobvious subject matter, then proposition 3 is false. Accordingly, the syllogism fails for valid patents.

Of course, propositions 1 and 2 are overblown generalizations, neither true nor false. A disclosure needn't be valuable to justify a patent, merely containing an enabling disclosure of a new, useful and unobvious. Proposition 2 requires some definitions before it can be analyzed. Even in view of those problems, however, of course the leap to the conclusion defies logic.

--
just another view,
Andy Greenberg
(remove "nonspam-" to get my real mailing address)

Subject: Re: Pamela Samuelson panelist at Open Source town meeting 8/21/98 San Jose, CA
Date: Mon, 10 Aug 1998 13:46:01 GMT
From: johnny_squire@hotmail.com
Message-ID: <6qmtip$151$1@nnrp1.dejanews.com>
References: <werdna-0608982217020001@tstpa1-110.gate.net>
Newsgroups:
misc.int-property,misc.legal.computing,comp.os.linux.misc,comp.os.linux.misc
Lines: 20

In article <werdna-0608982217020001@tstpa1-110.gate.net>,
werdna@nonspam-gate.net (Andrew C. Greenberg) wrote:

> If the disclosure is directed to an enabling disclosure of a new, useful
> and unobvious subject matter, then proposition 3 is false.

As ever, you've merely regurgitated the current practice of the profession. It's a question of fact, not law, whether a given legally enabling disclosure has actually taught anything to anyone.

> A disclosure needn't be valuable to justify a patent,
> merely containing an enabling disclosure of a new, useful and unobvious.

That's the current state of the practice. Is it right, good or constitutionally justified? No.

JS

-----== Posted via Deja News, The Leader in Internet Discussion ==-----
http://www.dejanews.com/rg_mkgrp.xp Create Your Own Free Member Forum

---

Subject: Re: Pamela Samuelson panelist at Open Source town meeting 8/21/98 San Jose, CA
Date: Mon, 10 Aug 1998 18:34:37 +0000
From: "Karsten M. Self" <kmself@ix.netcom.com>
Message-ID: <35CF3D3D.B21C8DF9@ix.netcom.com>
References: <6qmtip$151$1@nnrp1.dejanews.com>
Newsgroups:
misc.int-property,misc.legal.computing,comp.os.linux.misc,comp.os.linux.misc
Lines: 29

johnny_squire@hotmail.com wrote:
>
> In article <werdna-0608982217020001@tstpa1-110.gate.net>,
> werdna@nonspam-gate.net (Andrew C. Greenberg) wrote:
>
> > If the disclosure is directed to an enabling disclosure of a new, useful
> > and unobvious subject matter, then proposition 3 is false.
>

> As ever, you've merely regurgitated the current practice of the profession.
> It's a question of fact, not law, whether a given legally enabling
> disclosure has actually taught anything to anyone.

If the provision of IP protection by patent allows the inventor to publicize or describe his invention, then patent is promoting the advancement of knowledge. We don't need to show that patent claims are being read by other inventors (and they are), but that a patent claim provides security for the inventor to discuss his own invention without fear of losing rights to the idea.

--
Karsten M. Self (kmself@ix.netcom.com)

What part of "gestalt" don't you understand?
Welchen Teil von "gestalt" verstehen Sie nicht?

web: http://www.netcom.com/~kmself
SAS/Linux: http://www.netcom.com/~kmself/SAS/SAS4Linux.html

11:31am up 58 days, 9:00, 2 users, load average: 1.10, 1.20, 1.20

Subject: Re: Pamela Samuelson panelist at Open Source town meeting 8/21/98 San Jose, CA
Date: Mon, 10 Aug 1998 22:13:49 -0400
From: werdna@nonspam-gate.net (Andrew C. Greenberg)
Message-ID: <werdna-1008982213490001@tstpa1-41.gate.net>
References: <35CF3D3D.B21C8DF9@ix.netcom.com>
Newsgroups: misc.int-property,misc.legal.computing,comp.os.linux.misc,comp.os.linux.misc
Lines: 23

johnny_squire@hotmail.com wrote:
>
> In article <werdna-0608982217020001@tstpa1-110.gate.net>,
> werdna@nonspam-gate.net (Andrew C. Greenberg) wrote:
>
> > If the disclosure is directed to an enabling disclosure of a new, useful
> > and unobvious subject matter, then proposition 3 is false.
>
> As ever, you've merely regurgitated the current practice of the profession.
> It's a question of fact, not law, whether a given legally enabling
> disclosure has actually taught anything to anyone.

I dissent.

It's a question of fact whether a patent discloses new, useful and unobvious subject matter. If that is true, then it is apparent it must teach SOMETHING, since the subject matter is new, useful and unobvious as a matter of fact. If that is false, then the patent is invalid.

--
just another view,
Andy Greenberg
(remove "nonspam-" to get my real mailing address)

Subject: Re: Pamela Samuelson panelist at Open Source town meeting 8/21/98 San Jose, CA
Date: Wed, 12 Aug 1998 07:00:29 -0700
From: Bruce Hayden <bhayden@ieee.org>
Message-ID: <35D19FFD.25F5FA51@ieee.org>
References: <6qpiro$4jh$1@nnrp1.dejanews.com>
 <werdna-1008982213490001@tstpa1-41.gate.net>
Newsgroups: misc.int-property,misc.legal.computing,comp.os.linux.misc,comp.os.linux.misc
Lines: 48

johnny_squire@hotmail.com wrote:

> The difference here, and the thing I'm trying to point out, is that a patent
> may well be CAPABLE of teaching and _in fact_ teach nothing. Patent validity
> (capability) is a legal question. Whether any one human has learned from a
> given patent is a factual question. The latter SHOULD be the question that
> determines whether the patent system serves it's constitutional mandate.

But the proper question is not whether a given patent teaches, or really whether a given technology teaches, but rather whether the system as a whole teaches.

> I'm certain that whatever actual teaching, if any, software patents have done
> is an insufficient contribution to society to justify the monopoly granted.

First, you suggest that looking at the patent system on a technology by technology basis makes sense. It really doesn't. The patent law is written very broadly. It does not say that it covers everthing except these five technologies just because the tradeoff between disclosure and protection didn't work out positively for these technologies. Rather, pretty much all technologies are thrown in together when determining whether or not the beneifits of the patent system outweigh the costs.

Secondly, we are coming out of an era where really the only way

that software technology was taught was in printed publications. There was not any great wealth of patent information on software until very recently. But this was because for a long time it was difficult to get software patents. Plus, even during that time, many of the software patents were disguised as something else.

This of course has changed. An experienced software patent practioner can now overcome the nonstatutory problems of patenting software on a fairly consistent basis. As Greg A. in his Patent News statistice regularly points out, the number of software patents is rapidly increasing. We can expect that more and more of the prior art will be found, and taught in the patent data base.
--

The preceding was not a legal opinion, and is not my employer's.
Original portions Copyright 1998 Bruce E. Hayden,all rights reserved
My work may be copied in whole or part, with proper attribution,
as long as the copying is not for commercial gain.

Bruce E. Hayden bhayden@acm.org
Phoenix, Arizona bhayden@copatlaw.com
***Currently use the following: bhayden@uswest.net

Subject: Re: Pamela Samuelson panelist at Open Source town meeting 8/21/98 San Jose, CA
Date: Thu, 13 Aug 1998 08:18:48 -0700
From: Bruce Hayden <bhayden@ieee.org>
Message-ID: <35D303D8.4CDEEE48@ieee.org>
References: <6qsjs7$f6$1@nnrp1.dejanews.com>
<35D19FFD.25F5FA51@ieee.org>
Newsgroups: misc.int-property,misc.legal.computing,comp.os.linux.misc,comp.os.linux.misc
Lines: 36

johnny_squire@hotmail.com wrote:

> > First, you suggest that looking at the patent system on a
> > technology by technology basis makes sense. It really doesn't.
>
> I guess I should be flattered, but it's Prof. Samuelson's suggestion, not
> mine. FWIW, I think firmware embodiements of mathematical algorithms should
> be nonstatutory too, so it isn't a technology by technology thing - its a 35
> USC 101 thing.

Well, at least you are consistent. However, both are for the most part now patentable. Allapat basically said that a programmed general purpose computer becomes a special purpose computer, which is a machine under section 101. Note BTW that software per se continues to not be patentable (except possibly in a method claim). Rather, software in connection with some hardware is what is patentable. That connection may be loading it in RAM (a machine), or written to a floppy/CD (article of manufacture).

As for Prof. Sanuelson's position. First, as far as I am aware, it doesn't have any real legal basis. Secondly, she is not, and never has been either a programmer (her husband is, or rather was) or a patent attorney. And more importantly, I will suggest here that most of her observations were made at an earlier time, when there was much less software patent prior art in existence, and what there was was often hidden as electronics.
--

The preceding was not a legal opinion, and is not my employer's.
Original portions Copyright 1998 Bruce E. Hayden,all rights reserved
My work may be copied in whole or part, with proper attribution,
as long as the copying is not for commercial gain.

Bruce E. Hayden bhayden@acm.org
Phoenix, Arizona bhayden@copatlaw.com
***Currently use the following: bhayden@uswest.net

Subject: Re: Pamela Samuelson panelist at Open Source town meeting 8/21/98 San Jose, CA
Date: Fri, 14 Aug 1998 01:05:58 -0700
From: "Roger Schlafly" <nospam.schlafly@cruzio.com>
Message-ID: <6r0r00$o98$1@camel25.mindspring.com>
References: <6qvhunqjf1@nnrp1.dejanews.com>
<35D303D8.4CDEEE48@ieee.org>
Newsgroups: misc.int-property,misc.legal.computing,comp.os.linux.misc,comp.os.linux.misc
Lines: 29

johnny_squire@hotmail.com wrote in message
<6qvhunqjf1@nnrp1.dejanews.com>...
>Somewhere a bunch of posts back, this thread started as a discussion of
>(Samuelson's) sui generis protection v. patents for software. "Real legal
>basis" (beyond the constitution) isn't at issue. I know the current state
of
>the law, and appreciate the patient reiterations by the patent
professionals,

>but aren't you guys (Hayden, Greenberg, Aquilla, others?) willing to
discuss
>whether something better that what we have is possible?

Sure, something better is possible. I'd be all in favor of a sui generis software protection if I thought there was any chance of something reasonable being passed and implemented. But the suggestions from law professors (like Samuelson) is so silly, impractical, and out of touch with reality that it is laughable. And the stuff out of Congress is worse. The House just passed HR 2281 which is a sui generis protection for databases, and for copy protection. The law is stupid, ineffective towards its stated goals, downright harmful in its side effects, unreasonably favorable to the special interests pushing the bill, and generally bad from beginning to end. Every change that Congress proposes for copyrights and patents is bad. I see no hope that Congress could produce a sui generis software protection system which is better than what we have.

Subject: Re: Pamela Samuelson panelist at Open Source town meeting 8/21/98 San Jose,
CA
Date: Fri, 14 Aug 1998 07:18:31 -0400
From: werdna@nonspam-gate.net (Andrew C. Greenberg)
Message-ID: <werdna-1408980718310001@tstpa1-20.gate.net>
References: <6qvhunqjf1@nnrp1.dejanews.com>
<35D303D8.4CDEEE48@ieee.org>
Newsgroups:
misc.int-property,misc.legal.computing,comp.os.linux.misc,comp.os.linux.misc
Lines: 62

In article <6qvhunqjf1@nnrp1.dejanews.com>, johnny_squire@hotmail.com wrote:

> In article <35D303D8.4CDEEE48@ieee.org>,
> Bruce Hayden <bhayden@ieee.org> wrote:
>
> > Well, at least you are consistent. However, both are for the most
> > part now patentable.
> >
> > [true, but irrelevant statement of current law snipped]
> >
> > As for Prof. Sanuelson's position. First, as far as I am aware,
> > it doesn't have any real legal basis.
>
> Somewhere a bunch of posts back, this thread started as a discussion of
> (Samuelson's) sui generis protection v. patents for software. "Real legal
> basis" (beyond the constitution) isn't at issue.

I think he was talking [in sections snipped] about her other legal arguments about the patent system as it presently stands. In particular, Ms. Samuelson still writes from about a decade-old perspective concerning Section 101. The law has evolved considerably (actually gone back to where it was before Benson), but Professor Samuelson seems to be focused still on legal arguments not really meaningful in patent practice.

Agrees she advocates sui generis patent protection; agree that some of the discussion has been directed to that. Some of it hasn't. For example, hasn't Mr. Squire been making arguments in this very thread that the present regime isn't constitutional?

> I know the current state of
> the law, and appreciate the patient reiterations by the patent professionals,
> but aren't you guys (Hayden, Greenberg, Aquilla, others?) willing to discuss
> whether something better that what we have is possible?

That's interesting. I know few practicing lawyers in this area who would make that claim, not from ignorance, but from the fact that the "state of the law" does not reduce to bright-line rules easily scryed from cases.

And yes, we *ALL* consider whether something better is possible. However, this must begin with a full and fair analysis of what is wrong with the status quo; and a clear-minded recognition at what will likely happen if Congress ever decides to revisit the Patent Act in this regard. Current experiences in patent reform and the database legislation are, well, scary reminders of what sausage legislation can be. Additionally, changes to legislation for inventions rlated to software raise a host of serious consequenes and problems, not the lest of which is the decade of uncertainty as the new laws are construed by the courts.

This is, I agree, a conservative position: don't fix it unless its broken AND you know the fix will be better, and the mechanics will cluefully apply the fix correctly. However, I think we can all agree that there is a great deal at stake.

In this view, it seems to me responsible to assure that everyone is fully apprised about the current state of the law. In my eperience, most lay (and some lawyer) posters on the internet seem to be less than fully briefed in this arena.

--
just another view,
Andy Greenberg

(remove "nonspam-" to get my real mailing address)

Subject: Re: Pamela Samuelson panelist at Open Source town meeting 8/21/98 San Jose, CA
Date: Fri, 14 Aug 1998 06:58:09 -0400
From: werdna@nonspam-gate.net (Andrew C. Greenberg)
Message-ID: <werdna-1408980658090001@tstpa1-20.gate.net>
References: <6quom3lph1@nnrp1.dejanews.com>
<werdna-1208982256220001@tstpa2-80.gate.net>
Newsgroups: misc.int-property,misc.legal.computing,comp.os.linux.misc,comp.os.linux.misc
Lines: 52

In article <6quom3lph1@nnrp1.dejanews.com>, johnny_squire@hotmail.com wrote:

> In article <werdna-1208982256220001@tstpa2-80.gate.net>,
> werdna@nonspam-gate.net (Andrew C. Greenberg) wrote:
>
> > Doubtful distinction, IMHO. My Webster's Third New International (which I
> > take it is an adequate representative of the English language) defines to
> > "teach" to mean "to cause to know how to do something."
> >
> > Something that satisfies the stautory prerequisites for patentability
> > clearly accomplishes this. To the extent it does not, it clearly does not
> > satisfy the statutory prerequisites:
>
> Wrong. The statutory prerequisite is that the patent disclosure CONTAIN
> disclosure sufficient to teach.

Then we agree it teaches.

> The constitutional prerequisite is that the
> patent SHARE information in exchange for a monopoly. The law wrongly assumes
> that people skilled in the art will read and learn from the patents.

You won't find these words in the constitution. And if Johnny's sole point is that a book doesn't teach unless it is read, I really cannot quibble. My reply, however, is, "so, what?" Why wouldn't it teach anyone who reads?
>
> If a professor stands in front of an _empty_ room and lectures - there has
> been no teaching, even if the lecture was brilliant and capable of
> enlightening the masses. Teaching, by your definition above, requires that
> someone learn. Software patents are like the lecture to an empty room - no
> one hears what they have to say.

Seriously, Johnny, what is your point? That because YOU don't read patents, the patent system is unconstitutional? Nonsense.

It is certainly not true that NOBODY reads patents. Many people do. In fact, how many patents were cited in this very thread and the parallel threads as we are writing, with URL's directly to the IBM patent server? Anti-patent people are reading patents all the time (hoping to find another "bad software patent" to hype), and frequently publishing its text and text of its claims.

I rest my case. To the extent Johnny Squire's now agrees that the patents do teach so long as they are read by persons, I believe they serve their public purpose. Even if it were true that the public chose not to read them -- it wouldn't be much of an argument that they were deficient.

--
just another view,
Andy Greenberg
(remove "nonspam-" to get my real mailing address)

Subject: Re: Pamela Samuelson panelist at Open Source town meeting 8/21/98 San Jose, CA
Date: Fri, 14 Aug 1998 20:02:20 GMT
From: johnny_squire@hotmail.com
Message-ID: <6r254bhfq1@nnrp1.dejanews.com>
References: <werdna-1408980658090001@tstpa1-20.gate.net>
Newsgroups: misc.int-property,misc.legal.computing,comp.os.linux.misc,comp.os.linux.misc
Lines: 29

In article <werdna-1408980658090001@tstpa1-20.gate.net>,
werdna@nonspam-gate.net (Andrew C. Greenberg) wrote:
> Seriously, Johnny, what is your point? That because YOU don't read
> patents, the patent system is unconstitutional? Nonsense.

I do read patents, but I'm not a programmer anymore. Software patents today are problematic (not unconstitutional) because NOBODY OF ORDINARY SKILL IN THE ART reads them. Can you have three non-lawyer programmers step forward and claim to have learned something from a patent? I can't. I NEVER saw one when I made my living as a programmer. I certainly wasn't taught to use them as a resource in my Computer Science courses, nor in any of the developer training I attended.

I believe that my fairlure to learn anything from patent disclosures as a programmer is very typical (the rule?) and would shut up if someone showed me otherwise.

> It is certainly not true that NOBODY reads patents. Many people do. In
> fact, how many patents were cited in this very thread and the parallel
> threads as we are writing, with URL's directly to the IBM patent server?

The patents quoted in this thread have been discussed only because non-lawyer
programmers find them obvious or ridiculous, not because they have something
to teach.

JS

-----== Posted via Deja News, The Leader in Internet Discussion ==-----
http://www.dejanews.com/rg_mkgrp.xp Create Your Own Free Member Forum

---

Subject: Re: Pamela Samuelson panelist at Open Source town meeting 8/21/98 San Jose,
CA
Date: Mon, 17 Aug 1998 11:48:36 -0700
From: "Roger Schlafly" <nospam.schlafly@cruzio.com>
Message-ID: <6r9tp2ius1@camel25.mindspring.com>
References: <6r9q59$3il$1@nnrp1.dejanews.com>
<35D65EE8.67F4EFC2@ieee.org>
<6r254bhfq1@nnrp1.dejanews.com>
Newsgroups:
misc.int-property,misc.legal.computing,comp.os.linux.misc,comp.os.linux.misc
Lines: 29

johnny_squire@hotmail.com wrote in message
<6r9q59$3il$1@nnrp1.dejanews.com>...
>The difference is that until recently, S/W was nonstatutory. That's where
my
>argument stops.
>
>If the little guys (or the anti-S/W-patent guys of whatever size) in the
S/W
>industry had their act together, they could get Congress to do as you
>suggest. Heck, if doctors can make medical technique patents unenforcible
>against them, maybe can programmers do something similar. After all the
time
>they've spent on Y2K, programmer immigration and Microsoft IE, Congress
must
>have a new appreciation for the industry... ;)

That might be refreshing for Congress to at least look at the question.
The situation we have now has been dictated not by Congress, not
by the patent office, not by the supreme court, but by an inferior court
in Washington which is fond of ignoring precedent and which is
composed of judges who know nothing of software. Most of them
don't even have a patent law background.

The supreme court decision in Benson was not reversed in Diehr,
and is supposed to still be good law. Yet the federal circuit ignores it.

Subject: Re: Pamela Samuelson panelist at Open Source town meeting 8/21/98 San Jose,
CA
Date: Mon, 17 Aug 1998 20:32:28 -0700
From: Bruce Hayden <bhayden@ieee.org>
Message-ID: <35D8F5CC.936FAB76@ieee.org>
References: <6r9tp2ius1@camel25.mindspring.com>
Newsgroups:
misc.int-property,misc.legal.computing,comp.os.linux.misc,comp.os.linux.misc
Lines: 48

Roger Schlafly wrote:

> That might be refreshing for Congress to at least look at the question.
> The situation we have now has been dictated not by Congress, not
> by the patent office, not by the supreme court, but by an inferior court
> in Washington which is fond of ignoring precedent and which is
> composed of judges who know nothing of software. Most of them
> don't even have a patent law background.

And you think that J. Douglas did in Benson? The Federal Circuit
at least has some patent expertise. Congress and the Supreme Court
have essentially none. Indeed, if you read the cases coming out
of the Federal Circuit, it is clear that they know quite a bit more
about patents than you do.

As for understanding software, that is (IMHO) precisely why the trend
has been towards patenting, and not aways from it. Do you really
contend that J. Douglas in Benson understood the first thing about
software? At least today, most judges, and invariably all of their
secretaries and clerks know at least enough about software to run
their own computers. It is doubtful that most of the Court in
Benson had ever even seen a computer in person.

> The supreme court decision in Benson was not reversed in Diehr,
> and is supposed to still be good law. Yet the federal circuit ignores it.

No they don't. However, they have narrowed it quite a bit.

This is what I call wishful thinking law. You wish that software wasn't patentable, and thus it really isn't, despite the courts consistently deciding to the contrary.

Reality Check. The Federal Circuit has significantly narrowed Benson, taking the lead of the Supreme Court in Diamond v. Diehr. That is the law. Software is patentable if properly claimed. Sorry.
Arguing to the contrary, as you apparently are doing, is counter-productive. (Arguing that software isn't legally patentable is quite different from arguing that software shouldn't be patentable).
--

The preceding was not a legal opinion, and is not my employer's.
Original portions Copyright 1998 Bruce E. Hayden,all rights reserved
My work may be copied in whole or part, with proper attribution,
as long as the copying is not for commercial gain.

Bruce E. Hayden bhayden@acm.org
Phoenix, Arizona bhayden@copatlaw.com
***Currently use the following: bhayden@uswest.net

Subject: Re: Pamela Samuelson panelist at Open Source town meeting 8/21/98 San Jose, CA
Date: 7 Aug 1998 11:01:24 +0500
From: cmikk@tig.oss.uswest.net (Chris Mikkelson)
Message-ID: <35cb24d4.0@news2.uswest.net>
References: <werdna-0608982229040001@tstpa1-110.gate.net>
<35c8bd25.0@news2.uswest.net>
Newsgroups:
misc.int-property,misc.legal.computing,comp.os.linux.misc,gnu.misc.discuss
Lines: 32

In article <werdna-0608982229040001@tstpa1-110.gate.net>,
Andrew C. Greenberg <werdna@nonspam-gate.net> wrote:
[Regarding two "impossible" compression patents]
>I would be curious what language in the specifications of either patent
>that Mr. Mikkelson feels is directed to an impossible invention. I do not
>claim such language isn't present there, only that upon a cursory scan, I
>did not find it.

Jean-loup Gailly has done a more thorough examination than either of us.

http://w3.teaser.fr/~jlgailly/05533051.html
(it has a link to an analysis of the second patent, also)

Mr. Gailly is the author of gzip, the GNU compression program. He discovered these two patents while verifying that gzip did not use any patented compression algorithms.

While re-discovering J-l Gailly's page (I hadn't read it in a while), I found a link to the comp.compression FAQ. According to its section on compression patents, run-length encoding is patented!

I would certainly have suspected that to fail the "not obvious" requirement. Supposedly these patents were granted in '86 and '89 [warning: this post is not y2k-compliant], and I have a hard time believing that there is no prior art. I was under the impression that the run-length encoding idea was "as old as the hills", so to speak.

--
Chris Mikkelson | Economics is a zero-sum game. Anyone who says
cmikk@uswest.net | otherwise is selling something.

Subject: Re: Pamela Samuelson panelist at Open Source town meeting 8/21/98 San Jose, CA
Date: Fri, 7 Aug 1998 17:05:27 -0700
From: "Roger Schlafly" <nospam.schlafly@cruzio.com>
Message-ID: <6qg4ok$f5g$1@camel29.mindspring.com>
References: <35cb726a.0@news2.uswest.net>
<6qfqi6$l1t$1@jaka.ece.uiuc.edu>
Newsgroups:
misc.int-property,misc.legal.computing,comp.os.linux.misc,gnu.misc.discuss
Lines: 35

Chris Mikkelson wrote in message <35cb726a.0@news2.uswest.net>...
>I think that the central problem with Software patents really *is*
>the relative speed at which the computer industry moves. If you
>have a software idea, generally you can have a saleable product
>within a couple of months. Once you have the "prototype" (in the
>more traditional sense, i.e. a first working example of the final
>product), you have the product, and it costs next to nothing to duplicate
>it. Thus, the profits roll in much faster.

Dream on. Most significant new software products take 2+ years to develop, 6+ months to test, and millions of dollars to bring to market.

>Compare this to a more traditional, physical widget of some kind. After
>the invention, you must also manufacture a factory, to produce the items.
>Then, you must pay for the materials which the item is made out of. The
>manufacturing costs (the incremental cost-per-item) is much higher relative
>to the price people will pay, so it may actually take 17 years for you
>to make a decent profit.

There are drugs and other products which have relatively low
manufacturing costs.

>PS: On a related note, just imagine that Dan Bricklin had patented the
>spreadsheet. It was, I believe, a mostly original idea at the time. At
>least original enough for the patent office to accept ;-).

Actually, he could not have patented it because Pardo had
already patented it. The patent's validity was upheld in court,
but when the inventor sued Lotus, it was knocked out on a
trivial technicality.

Subject: Re: Pamela Samuelson panelist at Open Source town meeting 8/21/98 San Jose,
CA
Date: Fri, 07 Aug 1998 22:51:31 -0400
From: werdna@nonspam-gate.net (Andrew C. Greenberg)
Message-ID: <werdna-0708982251310001@tstpa1-18.gate.net>
References: <35cb726a.0@news2.uswest.net>
<6qfqi6$l1t$1@jaka.ece.uiuc.edu>
Newsgroups:
misc.int-property,misc.legal.computing,comp.os.linux.misc,gnu.misc.discuss
Lines: 57

In article <35cb726a.0@news2.uswest.net>, cmikk@tig.oss.uswest.net (Chris
Mikkelson) wrote:

> In article <6qfqi6$l1t$1@jaka.ece.uiuc.edu>,
> Steve Peltz <peltz@jaka.ece.uiuc.edu> wrote:
> >I think the reasoning is that, although the algorithm itself isn't
> >patentable, the application of the algorithm is (i.e. what is patented
> >is the idea of using the topological sort as a method of determining
> >the order for recalculating cells in a spreadsheet).
> >
> >Not that I think that's how it should work...
>
> I really have a difficult time seeing algorithms in anything other
> than a mathematical way.

Not a problem -- just don't sweat it. Mathematical or not, the
APPLICATION of an algorithm to a particular patent, expressed as an
apparatus, article of manufacture or method will be patentable. _See_
State Street Bank.

> I think that the central problem with Software patents really *is*
> the relative speed at which the computer industry moves. If you
> have a software idea, generally you can have a saleable product
> within a couple of months.

Decades old notion. It takes way more than that much time to print
T-shirts at today's vaporware rates. No serious software is developed in
this timeframe. Prototypes are not meaningful "products" in the sense
described.

> PS: On a related note, just imagine that Dan Bricklin had patented the
> spreadsheet. It was, I believe, a mostly original idea at the time. At
> least original enough for the patent office to accept ;-).

Actually, there were spreadsheet applications (both patent and software
applications) well before Bricklin. However, it is amusing you should say
this, as I just wrote a newsletter describing the State Street Bank case,
beginning with a discussion of Bricklin & Frankston's consultation with
their attorney just a few years before Diamond v. Diehr.

> Remember when VisiCalc came out? I was alive, but don't remember. Remember
> what the state-of-the-art computer was back then? The most advanced software?
>
> 17 years is just plain too long.

Yes, I do remember when Visicalc came out. I remember the absolute thrill
of being the author of the very first software product ever to beat it on
the SoftTalk monthly sales list.

But your conclusion is nonsense -- the notion of a spreadsheet in software
is as viable today as it was then, in many ways moreso. The invention is
as commercially important now as it was then.

--
just another view,
Andy Greenberg
(remove "nonspam-" to get my real mailing address)

Subject: Re: Pamela Samuelson panelist at Open Source town meeting 8/21/98 San Jose,

CA
Date: 10 Aug 1998 15:05:47 -0400
From: Craig Burley <burley@tweedledumb.cygnus.com>
Message-ID: <y6af5cwtbo.fsf@tweedledumb.cygnus.com>
References: <6qfn8t$sem@bourbon.cs.umd.edu>
<35C74BAC.547245AC@ix.netcom.com>
Newsgroups:
misc.int-property,misc.legal.computing,comp.os.linux.misc,gnu.misc.discuss
Lines: 140

jsm@mindspring.cob writes:

> Craig Burley (burley@tweedledumb.cygnus.com) wrote:
> > "Roger Schlafly" <nospam.schlafly@cruzio.com> writes:
> > > Most significant new software products take 2+ years to develop,
> > > 6+ months to test, and millions of dollars to bring to market.
>
> > The 1998-08-10 Forbes article on free software seemed to suggest
> > otherwise, especially about that last part.
>
> I want to agree with you, Craig, really I do, but I keep getting
> stuck on the real value of all that volunteer programming talent
> at (a conservative) $50/hr.

In the original context -- software patents vis-a-vis the pace
of the software industry -- do you think that value is pertinent?

Put another way: if I plan to spend $2M to create a new, proprietary
Fortran compiler, perhaps it can be argued that I need software
patents to preserve my investment until I get significant ROI from
sales of individual copies. I can allocate a suitable portion of
that $2M up front to code reviews for patentable matter, hiring of
attorneys, and so on. No problem.

OTOH, if you claim that $2M is a reasonable approximation to what
it took to create g77, a widely used free Fortran compiler -- or
will take to create it, depending on how you define it as "finished" --
then I'd be happy to agree with you as far as *that* goes.

However, you'd also have to show that it is about as easy to allocate
a portion of that $2M for *real* up-front spending on lawyers, patent-
application fees, and other legal mumbo-jumbo, which is doggone
hard to do when that $2M is *not* real money -- just value imputed
by you (and many others, of course) *after* the project is complete,
or substantially so.

If you can't do that, perhaps that suggests something important
vis-a-vis whether the patent system is a help, or a hindrance, to
software development in *general*, given that at least one major
(and growing) methodology used is *hindered* by the patent system.

I believe it *is* a hindrance, and an especially nasty one if it
can't keep up with relatively minor, short-term changes (compared
to other long-patentable fields) like the free-software movement
upon which more and more businesses are coming to depend.

So my next question would be: exactly what purpose would software
patents serve that a free-software product like g77 would *need*?

So far, my answer is: the only possible use is defense against
large corporations already using software patents to prevent certain
kinds of free software from competing with their proprietary products.
That is, if software patents don't exist, or are sufficiently
restricted so that general-purpose computing is rarely entangled
within it, then there's no need for them in free-software products
like g77.

The original quote to which I responded seemed to be saying "no
way the software industry goes too fast for the patent system,
all the major packages take forever and tons of money to develop",
and the Forbes article seems to say "not really" to at least some
of those assumptions.

If you want to *value* the efforts taken to *rapidly* develop
free software at some dollar value, fine, but you might as well
just say "well, instead of taking 10 programmers 2 years to create,
it took 120 programmers 2 months to create, which amounts to the
same amount of *overall* time".

Would you then claim that the patent system is still sufficient
to keep up with *that* pace, on the theory that man-hours are
man-hours however they're implemented in "wall-clock time"? How
far would you be willing to go down that road -- claim that a
10-year-average approval process is "okay" because, after all,
it'd take one programmer *20* years to develop the average
major application, so having the approval process take only "half
the time" it takes for a typical major project to be brought
to market, if done, *theoretically*, by only one programmer,
does not constitute a hindrance regardless of how many programmers
are actually thrown at the job and thus how little time it
actually takes?

In summary: the present patent system is entirely built around the
concept of *up-front* development and related legal work, including
the concomitant spending and, thus, allocations of cash. The

burgeoning free-software movement is entirely built around the concept of spontaneous, often "bushy", generally unplanned, growth requiring little cash and thus little or no initial allocations.

Therefore, software patents are unlikely to be anything but a substantial hindrance to the free-software development paradigm, as it's clear they're of no fundamental help. (Free software is inherently published, as source code, so any time taken to find patentable inventions in it is most efficiently used by simply publishing such things forthwith, rather than submitting them for a long, "secret" review by the PTO. Note: I believe some non-US PTO's don't keep them secret, but have lost track; these might pose somewhat lower hindrances to free-software development for that reason.)

Of course, we (in the free-software community) have known that software patents pose a danger for *many* years. The one thing that might change this equation somewhat is if a bunch of really bright lawyers "convert" to a similar paradigm, a free-IP one, helping their "brothers-in-arms" in the free-software community by rapidly applying for relevant patents for use by that community and, at the same time, challenging software patents that are held by proprietors. That'd mitigate, somewhat, the up-front cash-spending needs, for lawyers anyway -- though us programmers don't have to spend thousands of dollars anytime we want to publish some new code the way lawyers do to file or challenge a patent, again illustrating the impediments the patent system places in the way of the coming Information Age.

(As far as whether a lawyer might get the some "buzz" by successfully challenging a patent, taking on all costs himself and getting no rewards except pats on the back, as some of us programmers get when we find and fix a big bad bug in a popular application under similar risk/reward circumstances -- I can't speak to that, as I don't know enough lawyers.)

For the most part, this is all an interesting theoretical exercise. At some point, there might occur some pretty serious examples of how much of an impediment software patents are to the entire industry.

E.g. Microsoft might decide to attack Linux on patent-violation bases, or some such thing. This is one possible scenario that might bring about substantial industry-wide antagonism towards software-patent proprietors, just as the possibility of widespread Y2K outages worldwide might conceivably bring about substantial antagonism towards anyone who distributes software without source code for public review.

Another summary: since the patent system is all about ensuring that inventions are published, it can offer *no* advantages to the free-software community, since that community *necessarily* publishes all of its inventions, in the form of source code available for public review.
--

"Practice random senselessness and act kind of beautiful."
James Craig Burley, Software Craftsperson burley@gnu.org

Subject: Re: Pamela Samuelson panelist at Open Source town meeting 8/21/98 San Jose, CA
Date: Mon, 10 Aug 1998 22:47:20 -0700
From: "Roger Schlafly" <nospam.schlafly@cruzio.com>
Message-ID: <6qolo6$2a$1@camel25.mindspring.com>
References: <werdna-1008982237130001@tstpa1-41.gate.net>
<y6af5cwtbo.fsf@tweedledumb.cygnus.com>
Newsgroups: misc.int-property,misc.legal.computing,comp.os.linux.misc,gnu.misc.discuss
Lines: 22

Andrew C. Greenberg wrote in message ...
>> E.g. Microsoft might decide to attack Linux on patent-violation
>> bases, or some such thing.
>
>They might, they just might. So far, however, Microsoft has been the
>world's leading VICTIM of software patents, being on the losing end of the
>STAC lawsuit to the tune of $100M. There's no other meaningful litigation
>in this arena. In other words, software patents have benefitted Davids
>far more than Goliaths so far as support can be found in the public
>record.

This analysis is incorrect. First, Stac settled for an amount much less than $100M. Second, Microsoft lobbies in favor of strong software patent, and it isn't known for acting against its own self interest. Third, patent license negotiations are almost never on the public record. Fourth, software patents raise the barrier to entry in the software market, and that fact works to Microsoft's favor.

Subject: Re: Pamela Samuelson panelist at Open Source town meeting 8/21/98 San Jose, CA
Date: 11 Aug 1998 15:15:21 -0400

From: Craig Burley <burley@tweedledumb.cygnus.com>
Message-ID: <y6hfzjl48l.fsf@tweedledumb.cygnus.com>
References: <6qfn8t$sem@bourbon.cs.umd.edu>
<35C74BAC.547245AC@ix.netcom.com>
Newsgroups:
misc.int-property,misc.legal.computing,comp.os.linux.misc,gnu.misc.discuss
Lines: 529

werdna@nonspam-gate.net (Andrew C. Greenberg) writes:

> > However, you'd also have to show that it is about as easy to allocate
> > a portion of that $2M for *real* up-front spending on lawyers, patent-
> > application fees, and other legal mumbo-jumbo, which is doggone
> > hard to do when that $2M is *not* real money -- just value imputed
> > by you (and many others, of course) *after* the project is complete,
> > or substantially so.
>
> Nothing like $2M is ever necessary to acquire patent protection for an
> invention. And if you are relying upon grant money from a University,
> there is always that resource. If you are relying upon volunteer work for
> the benefit of the public, you might seek a lawyer to work pro bono
> publico (lawyerese for "for the benefit of the public") to help you
> protect it. If you can't find someone to do it pro bono, that might
> suggest something about the public benefit of the work.

I'm sorry, but could you miss the point *more* than you do?

The ORIGINAL QUOTE was about LARGE software projects, and was in response to the claim that the patent system might be a hindrance to software development. AFAIK, *each patent* costs $K to apply for, with no promise it'll get approved, so for even medium-scale, $200K development projects, the cost of patent applications needed to even *try* to provide a defense against litigation by other patent-holders can consist of well beyond 5% of the total cost, assuming that cost is *cash* (which is not the case for free software -- the cost there is, instead, value imputed to the labors of volunteers).

You're just making the excuse that the free-software community can *theoretically* afford to fully engage the software-patent system so that you don't have to defend that system against quite-legitimate charges that it serves largely as a hindrance, not a help, to software development *in general*, and this will get *worse*, not better, as free-software development gains prominence, since such development *practically* cannot afford to "slow down" just to cope with software patents.

Further, individuals like myself *cannot afford* to do our volunteer development under an ever-increasing threat of litigation over software patents. No amount of theoretical hand-waving will change this. Only the elimination of software patents can assure that it won't be a problem. The likely solution will be some middle ground, e.g. something like what Karsten talks about.

The fact is, I very nearly stopped writing g77 because of the threat of *personal* exposure vis-a-vis software patents, and decided to plow ahead only because it seemed unlikely that a compiler for FORTRAN 77 would violate any important patents *or* that the Fortran community would tolerate a lawsuit against a volunteer writing code supportive of that community. I could *not* have gotten up-front funding to apply for software patents of my own, and even if I had, I couldn't have afforded to spend all the extra time to try and divine patentable material from the largish, but fairly mundane, code I was writing at the time -- code containing inventions that would be published *anyway*, and thus did not need the patent system to be "teased out" of my brain.

More cutting-edge software products are starting to be written by free-software developers, and the only way software patents won't be purely a hindrance to these is if the computer industry as a whole refuses to sue, or threaten to sue, developers and distributors of free software for patent violation.

However, this would require a change in behavior, since the industry has *already* so threatened, just as the Linux name was "tied up" in ludicrous litigation because some bozo decided he could trademark the name, even though on the *face* of it, he had no case. Linux "won", as it should have, of course, but it took time *and* money to bring about the obviously correct result.

In the case of software patents, instead of missiles (as in trademarks), which are clearly visible as they approach and forthrightly dealt with, we have land mines, planted years earlier in the hopes of catching someone. The perceived value they're likely to have, especially among the "if-it's-legal-it-must-be-ethical" crowd that, so often, includes substantial numbers of lawyers, will not necessarily exclude free-software developers among their targets.

As these mines begin detonating, people will take note and free-software development will slow down, with no benefit accruing to the industry as a whole, since all free-software development is published *anyway*.

On the plus side, I thought this would have happened by now, but
am unaware of any substantial increase in such cases. However,
Netscape probably wasn't worried about anyone giving away free
knock-offs of its products, until it started actually happening.

> > If you can't do that, perhaps that suggests something important
> > vis-a-vis whether the patent system is a help, or a hindrance, to
> > software development in *general*, given that at least one major
> > (and growing) methodology used is *hindered* by the patent system.
>
> I don't think I agree. Property rights in general make it difficult for
> poor authors to write as much as they would like, instead needing to work
> to feed and house themselves and their families; they make it difficult to
> perform theatrical productsions, it being difficult to find resources such
> as a theatre and advertising; they make it difficult to make movies, yet
> propery generally works well for the benefit of society. Some things cost
> money to do, others do not. If what you are saying is that it is
> difficult to write free software for free without having the benefit of
> legal counsel (or a computer, or electricity), I agree. Does that mean
> the patent system is broken, or that "the software industry" is hindered
> thereby? I don't think so, any more than the complaint that you need to
> have a computer to write software makes the personal property system a
> hindrance. Still others can get jobs doing software for the same reason
> the freebies are having problems.

Again, you're missing the point. Nobody in this sub-thread was talking
about the validity of IP (or property rights) in general.

The assertion was that software patents have *too high* a cost.

The response was, essentially, "not compared to large software
projects, which take many programmers 2+ years and $2M", at which
I chimed in about the pertinence of that to the entire industry
given the rise of free-software development (for which *imputed*
value clearly has nowhere near the usefulness of *actual cash*
when it comes to funding patent applications and legal defenses).

Is it totally unacceptable to you to discuss whether software patents
have *too high* a cost, and perhaps should be special-cased, if not
eliminated, within the patent system, *without* taking an all-or-
nothing stance with regard to all IP, or even all property rights??

If so, please leave the discussion, as you can't handle it. We're
not discussing an all-or-nothing choice; we're discussing the
quite-legitimate question of whether software patents are more
trouble than they're worth. That's exactly what citizens like
us are *supposed* to be doing, right?

> > I believe it *is* a hindrance, and an especially nasty one if it
> > can't keep up with relatively minor, short-term changes (compared
> > to other long-patentable fields) like the free-software movement
> > upon which more and more businesses are coming to depend.
>
> It hasn't hindered anything meaningfully yet, and indications are that it
> has helped quite a bit. Once again, this seems to be a "Patents are bad"
> argument, without any particular reason to distinguish software patents
> from other works and other industries and other forms of IP.

Software patents have *destroyed* businesses. The "Bingo" patent
alone was fought, successfully, *only* because the owner of the
business that was destroyed decided to fight it *on principle*,
even knowing his business was already dead.

It did not matter that the patent and/or the patent owner overreached.
The beauracracy's response *necessarily* included the total destruction
of a viable business that involved *proprietary* software, because
that's basically how the system works -- this "David" fought, "won",
and lost his life, while Goliath got a dent in the metal helmet given
to him free by the government.

I have yet to see *any* substantial evidence that software patents
have "helped" much of anything, in terms of bringing products to
market. In terms of publishing software algorithms that would
otherwise not be published, it might have helped, but that's supposedly
not "allowed". In terms of publishing information on the areas
in which algorithms can be applied, I've not seen any evidence that
it has helped at all, and it *will* significantly hurt the entire
industry within ten years, *noticably*, as software itself makes
choices regarding what algorithms to apply to new problem domains
without the awareness of the humans posing the problems.

As far as software patents vs. other types: the distinctions are
sufficiently obvious for anyone wishing to objectively consider
them. Very few people have had trouble with this, although a few
who *discuss* "software patents" have insisted there are no real
differences. The LPF and others have published definitions that,
as with all legal definitions, are not precise, but sufficiently so that
they agree on the categorization of the bulk of patents.

> > So my next question would be: exactly what purpose would software
> > patents serve that a free-software product like g77 would *need*?
> >
> > So far, my answer is: the only possible use is defense against
> > large corporations already using software patents to prevent certain
> > kinds of free software from competing with their proprietary products.

>
> This isn't enough?

In other words, you are saying "software patents are *great* for the free-software community -- without them, how would that community defend itself against litigation based entirely on software patents?"

I think others, if not you, will see how ludicrous your circular argument is. *We* don't need software patents. The *world* does not need software patents or, if it might, it does not need them to apply to any free-software products.

IMO, the solution is simple: pass laws that rule out *all* software licensed under conditions that meet Open Source (TM) (or similar) guidelines, such as the GPL, as targets of any software-patent-related litigation.

In other words, since an entire subclass of a functional industry *always* publishes its inventions, don't allow it to be threatened by a beauracracy that exists solely to ensure inventions are being published.

(In reality, the beauracracy exists to provide instant riches for litigous people and generous fees for IP lawyers, which is why software patents will be defended to their death, if necessary. Of course, neither of these things truly benefits society.)

> > That is, if software patents don't exist, or are sufficiently
> > restricted so that general-purpose computing is rarely entangled
> > within it, then there's no need for them in free-software products
> > like g77.
>
> True. Others of us would like there to be products besides g77.

Name a *single* Fortran compilation system that could not be, or have been, released if software patents did not exist, and back up your assertions with evidence.

> > Therefore, software patents are unlikely to be anything but a
> > substantial hindrance to the free-software development paradigm,
> > as it's clear they're of no fundamental help.
>
> Now, we move away from the notion of the "software industry," to a pure
> "free-software deelopment paradigm."

No, we do *not* "move away" from that. That's where I *started* this subthread, based on a quote that suggested it did not matter how slow the software-patent bureacracy was, since So Much Money And Time was needed to develop software.

It's the free-software community that is *shaking up* the status quo, such as readers of Forbes, by *proving* that software need not take So Much Money And Time to develop.

Those of you who are in love with software patents don't care, because you're happier if software development is slowed down and made more expensive, as long as some of you can skim off the top. *All* beauracracies are loved and defended, fundamentally, for this reason.

I think the larger *user* community, desperate for rapidly developed, high-quality software, will have a *different* opinion, and needs to be informed about the issues.

> Lawyers *DO* enjoy legal work, or they should get out of it. I for one
> LOVE what I do (just as I loved what I did when I was writing and
> publishing computer games). The trick is not to hope someone would enjoy
> that, but to get some lawyers to start believing your religious views
> about the public benefits of the particular free software you want to have
> supported. Pro bono legal work happens all the time -- I do a great deal
> of it myself.

I said *nothing* about religion. The entire industry is coming more and more to depend upon the rapid, insubstantially funded free-software development efforts.

That a few of you get rich from software patents, thus causing you to support them no matter how much they slow down software development, will not prevent the *rest* of us from pointing out that that's exactly what you're doing.

When trolls take control of a bridge and charge tolls, most people accept them at first, because they're usually cheap. As the tolls rise, people usually continue to accept *that*, because the rise usually means the trolls have correctly recognized the increased need for the bridge.

At some point, however, one of two things can happen. A better, new bridge can be built somewhere that is more efficient for everyone, but troll-free (and perhaps toll-free), to keep its efficiency at full tilt. Or, the local markets that depend on the efficiency of the bridge can have the trolls removed, by force if necessary.

The free-software community is, right now, offering a significant

"new bridge" to meet the efficiency needs of the coming century.
It won't offer the *only* hope, perhaps, but it's clear it might
offer a substantial portion of the solution.

But those of you who plan on charging tolls on *this* bridge had
better be prepared to be barred from entering the community,
and/or "removed" upon attempting to slow it down by skimming
off the top. We don't *need* you to help us publish inventions,
since we already do so by necessity, after all, so any possible
benefit of trolls is wasted on us.

> One way to get lawyers interested is to stop bashing them and the system
> they address; particularly since the chances of changing the law as
> applied is virtually nil.

Isn't it amazing how lawyers are always the ones who get to *make*
the laws, then afterwards claim "well, you can't change the laws,
so don't bother trying -- just make friends with *us* and maybe
we won't charge so much for you to conduct business in this
environment"?

Sorry, but I don't buy it. The legal environment can and will be
changed to suit the goals of *society*, not *lawyers*, despite
what your current hegemony tells you.

Others have posted some arguments about the degree to which *effective*
copyright protection exists for software, with some claiming this
has gone downhill, so that proves the law *can* be changed -- and
I doubt this has been the desire of the bulk of IP lawyers, even
though I know at least one or two who found some of the growing
claims to be, at least, worrisome (usually because they happened to
be working for a company that'd be *targets* of litigation).

I'm not up on the copyright-lessening details, but I do know that anyone
who can't defend software patents applying to free software with any
justification beyond "they help to defend free software against
software-patent litigation" is likely to be on the wrong side of
how the law, and industry and society as well, will be changed to
accommodate the inherent efficiencies of free software.

(The "inherent efficiencies" I refer to come from the large-scale,
open sharing of source code. Whether such sharing is a "good thing"
in a philosophical, social, or religious sense is a completely
different topic, upon which I indeed have some views. I'm
referring to efficiencies that *no other form* of software development
can possibly realize, efficiencies that are quite clear to many of
us who've been doing all sorts of software development, including
both proprietary and free, for decades. They are objectively
reasoned about, instantiated, and measured. But they are not
the topic of these posts; the Forbes article hinted at some of them.)

> > For the most part, this is all an interesting theoretical exercise.
> > At some point, there might occur some pretty serious examples of how
> > much of an impediment software patents are to the entire industry.
>
> That leap back up from the "free software" paradigm to the "entire industry."

Exactly, since my original point was that the free-software paradigm
brings the promise of substantial solutions to problems the entire
industry is facing, at *least* in niche areas non-free development
just doesn't reach, but perhaps in areas such as OS, compiler,
and tailored-user-interface API and implementation development as well,
among many other possibilities.

And anytime people hand-wave the slow patent system as applied to
software because *traditional*, closed, proprietary development is
slow and expensive by *contrast*, we will try and quickly point
out that that development model is *not* the globally optimal model
for software development. (It would be *amazing* if we had managed
to collectively converg on a global optimum in a field that is,
basically, no more than about 20 years old. Yet some people act as
if we had, and claim that, because we've somehow already hit the
optimal system for meeting market needs, *any* litigation and/or
beauracracy should be above criticism as long as it doesn't
significantly impinge on that *particular* system, even though it
might hurt other, possibly more globally optimal, ones trying to
gain a foothold.)

> There's no other meaningful litigation
> in this arena. In other words, software patents have benefitted Davids
> far more than Goliaths so far as support can be found in the public
> record.

Name one case where a free-software product has *benefitted* from a
software patent against any Goliath at all. (Not that it might not
happen in the future.)

There are, of course, cases where free-software products have had to
be rewritten, or removed, due to patents. E.g. `compress' had
to be removed and tons of files using it re-compressed using a
new algorithm.

Of course, at this point the usual retort is "but this means the
software got *better* thanks to patents". Fine, I'll show up at
your house randomly and blow holes in it with a shot-gun, let me know

whether you agree that this'll be a *good thing* because it'll
make your house necessarily more secure (because you'll have *no
alternative* but to make it more secure, unless you just give up
and move out).

Oh, right, you're a lawyer: I'M JUST KIDDING.

In the meantime, the initial replacement for `compress', `gzip', is
now being superceded in some circles by `bzip2', yet it didn't
seem necessary for there to be a patent-infringement threat to make
that happen. What *was* necessary was lots of worry, research, and
delay to avoid too many people becoming dependent on a new compression
algorithm only to find out, later, that somebody *else* had already
patented it.

We've been burned before, and we're *already* being slowed down by
simply *avoiding* being burned again.

Permit us the luxury of discussing whether we'll continue to allow
our government to provide you patent lawyers with affordable flamethrowers.

> > Another summary: since the patent system is all about ensuring that
> > inventions are published, it can offer *no* advantages to the
> > free-software community, since that community *necessarily* publishes
> > all of its inventions, in the form of source code available for
> > public review.
>
> Note we are shifting back down to the "free-software community" (this is
> getting tiring -- pick one constituency and stay with it).

I picked one *in the beginning* of my entry to this thread, precisely
to illustrate how the arguments in favor of a big beauracracy that
can only slow software development don't apply *now*, and probably
will apply even less in the *future*.

Try *reading* the thread before posting.

> Here, however,
> I believe the author is misguided. The free-software community has not
> invented EVERYTHING it uses. Indeed, most of the algorithms used have
> come from without that community, using technology and algorithms learned
> and disclosed from others (most, but not all of which came from textbooks
> and other non-patent literature). Along with the rest of society, the
> free-software community will benefit (and will benefit moreso in the
> future) from the disclosures of inventions learned, directly or
> indirectly, from the incentives to disclose in the patent system.

We don't need, and have *never* needed, the patent system to make such
disclosures.

After all, we publish the source code, and so does everybody else in
this community. That's its defining feature.

Do you really think that, without the patent system, *no*
disclosures of substance would occur? Even in the free-software
community, where everything is published as *source code*?

How open are you to the idea of having *legal* practices, arguments,
and *decisions* be patentable subject matter? How would you like
to have successfully argued a case in court, won, collected your
fees, and later be successfully sued for millions of dollars because
it turns out somebody was awarded for a patent on the arguments
you used, for which they'd applied earlier or, perhaps, even
later? How would you like to have to tell a client "well, I'd
like to argue it *this* way, but I can't, because somebody else
has patented that method of argument and charges way too much for
us to use it"?

How would you like to have to spend a substantial portion of every
day asking yourself whether some approach to your legal duties
could be patentable material and, if so, whether it was worth
trying to spend the thousands of dollars to patent it, not to
mention the time and effort needed to write it up, all because
you were just trying to construct a possible defense against
actual litigation in the future? And still not be sure whether
any of those patents would be granted, and even if they were,
whether they'd be useful in any *actual* litigation brought
against you, in that what you had, the litigants didn't need?

Would the legal system become *more* or *less* efficient, given
such a scenario? What would happen vis-a-vis pro bono work, and
the ability of the poor to obtain competent legal counsel, for
civil, and for criminal, cases?

How much better would the world be if phrases like "Clinton's
patented spin machine", "Agassi's patented down-the-line backhand",
and "Barry Sander's patented cut-back" were *literally*, i.e.
legally, true?

Yet *all* of these activities take far more up-front analysis and/or
practice than the creation of typical free-software "inventions"
(in the patentable sense). That is, software is *more* fluid
than any of the above, because (among other things) it can be easily
and rapidly shared with a huge number of people around the world,
even when incomplete.

You can't do that with your legal arguments, because you can't give
away your strategy before us. Same for Clinton. Either you have to
keep your strategy secret until it is used, or, like Agassi and Sanders,
you have to practice and practice, plus learn from experience, to
discover what happens to work best, before you can really share
anything useful with anybody else, especially a very wide audience.

> With all due respect to Messrs. Stollman and his growing progeny, if the
> works of the inventions set forth in all that source code were all that
> the public knew and could know about software inventions, the world would
> be a poorer place, indeed. The other paid communities contributed a great
> deal toward the state-of-the-art as well.

There is a *trivial* number of inventions that would not have been
public knowledge without the existence of software patents, AFAIK,
encompassing the *entire* free-software source-code base. The
"software patents" I've heard about going back decades were all
on techniques that were *inherently published* with the code for
the system, so they hardly needed patent protection to persuade
their being published in the first place. And I doubt you'll be
able to go back and show that, without the ability to specifically
patent those "inventions", the products themselves (or the specific
inventions) would not have been brought to market and thus inherently
published anyway.

I worked for a variety of proprietary-software companies in the past,
and *never* did I hear about a software patent being a reason to
bring, or upon failing to be approved, *not* bring, an invention or
product to market. Mostly, we just heard about how we should try
to "discover" inventions that might be patentable in our code (which
was mighty hard, we were too busy writing it and getting it to work),
and how a patent would be a feather in our, and our employers', cap --
very little emphasis was placed on the use of such a patent for
actual litigation until the early '80s, when "software patents" began
to take on some real teeth.

> I think there is a middle ground to be found, and Karsten has put his
> finger on some very useful and key ideas to that end.

I agree, and those ideas are likely to be what will actually happen
in the short-to-mid term.

In the long term, society gains no benefit from software patents as
applied to free-software (or Open Source (TM)) products, so it is
best that they should not be applicable to them at all, legally or
otherwise.

If the non-free developers have to be protected against free
developers with software patents, then they have already lost
the battle -- and perhaps their best option is to throw off
the shackles of their *own* patent protection to better compete.
(As a sometimes-proprietary developer, I think that would be best.
I do have some reservations about whether society would need the patent
system for software run as *services*, e.g. via Network Computers,
which is thus never actually published, and is even more of a
problem than software published only as difficult-to-reverse-engineer
binaries in terms of sufficing.)
--

"Practice random senselessness and act kind of beautiful."
James Craig Burley, Software Craftsperson burley@gnu.org

Subject: Re: Pamela Samuelson panelist at Open Source town meeting 8/21/98 San Jose,
CA
Date: Wed, 12 Aug 1998 10:10:04 -0400
From: werdna@nonspam-gate.net (Andrew C. Greenberg)
Message-ID: <werdna-1208981010040001@tstpa2-11.gate.net>
References: <y6hfzjl48l.fsf@tweedledumb.cygnus.com>
Newsgroups:
misc.int-property,misc.legal.computing,comp.os.linux.misc,gnu.misc.discuss
Lines: 259

Craig's extensive posting is only summarized below, as extensive
quotations from his posting would make this thread burdensomely long. I
anticipate that he will feel from time to time that I misrepresented his
position, and apologize in advance: there was no way to answer
point-by-point without enormouse bandwidth expense.

Craig feels I missed his point. I don't think so, but it is clear he has
not responded to mine. I feel he has made several errors in his
reasoning:

* Free-software equates to "the entire software industry"

Craig accurately observes that the traditional economic incentives
provided to commercial businesses and individuals by the patent system do
not accrue to the benefit of a volunteer who claims no interest in a free
software product. This is undeniable. He then proceeds to observe that
the existence the patent system nevertheless adds costs and expense to the
process of developing all software, and hence also free software. This,
too, is undeniable. By simple arithmetic, he concludes that there is a
net loss under the patent system for free software.

On purely economic terms -- I do not disagree. Of course, if the free software movement, like public television and radio, were to capitalize on the system and fund some (or all of) its efforts through acquisition and licensing of strong IP, including patents, to non-free-software industry and individuals, the net economic measures would change.

He then makes the leap to conclude that the software patent system is bad for free software folks. This ignores the inchoate benefits to society of the patent system, of which even free software people are a part as well as the benefits of the disclosure that the patent system fosters, which even free software people can read.

In maintaining this position, he insists that economic arguments are missing the point, since the economic arguments do not benefit the free software folks.

From this conclusion, he makes the even broader leap to conclude that the badness for free software folk now extends to the "entire software industry." (He repeatedly uses the word "entire.") Without more, this point is indefensible, for the economic arguments he rejected *DO* apply to the entire software industry.

* The patent system is unavailable to the free software industry

This is misguided as well. Karsten and myself have both suggested ways in which low-rent, but effective use of the patent system can be obtained. Both involve obtaining the patronage and sponsorship of others, but that is, after all the point. If free software is a benefit to society (as many people believe public radio and television are a benefit), then it will make the effort to obtain such patronage. Just as volunteer effort, as well as housing and hardware are provided, so too can the patronage of industry and lawyers. If it can't, this informs the debate as to the social merits of free software.

* Software Patents vs. Patents at large

Craig claims he has not raised questions as to the benefits of IP in general, but merely focused on software patents in particular. Yet the great majority of his patent arguments fall into the categories of

"patents are bad" or

"bad software patents are bad."

The former, I *will* defened to the death (believing as I do in the overall benefits of the patent system), the latter I will not defend at all, believing likewise that bad software patents need to be exterminated, both by discouraging their gratuitous asssertion against less monied persons and by preventing their issue at the outset. I will note that "bad patents" are often used as an argument against the patent system -- bad patents other than software issue and are asserted all the time, yet somehow the American economy has weathered the storm -- it is true that there are often tragic consequenes in individual cases, just as there are when a down-on-their-luck family gets ejected from their home upon foreclosure by a bank unreasonably unwilling to work with them -- but such cases do not on balance justify abrogation of property rights because of the general benefits of property law to society as a whole).

But these arguments (both his and mine above) don't really get to the point. Why would it be so that

"software patents are bad"?

Why is software different? To this (and I will quote here becaue I do not want to presume to understand what he meant), Craig wrote:

> > As far as software patents vs. other types: the distinctions are
> > sufficiently obvious for anyone wishing to objectively consider
> > them. Very few people have had trouble with this, although a few
> > who *discuss* "software patents" have insisted there are no real
> > differences. The LPF and others have published definitions that,
> > as with all legal definitions, are not precise, but sufficiently so that
> > they agree on the categorization of the bulk of patents.

Clearly this is not a response. Craig excoriates me earlier (indeed suggesting I simply leave and not participate in the discussion) for failing to address what he asserts is "the subject" of the thread and not being responsive to his postings, yet his only substantive argument on the only substantive cases against software patents is that the answer is obvious. Further, it clearly does not matter how many people "believe" in the true faith -- even one lone dissenting person can be right!

I have addressed LPF's argument in this thread and others. Craig does not respond to those remarks either.

And let me make it clear. I do not merely "*discuss* 'software patents'." I practice in this area of law -- I know what is and isn't possible, and I know what is and is not inherent in the cost of obtaining a patent. I prosecute as eell as litigate patents, and understand what is going on here. I believe there are substantial misunderstandings on Craig's part as to these fundamental points, and will be pleased to enumerate these further, if he is inclined to listen. Further, I do not merely *discuss* software development. I have been a programmer, analyst and software designer for decades before I was a lawyer. I authored an

published best-selling computer games. I also worked in academia developing software. I *KNOW* what it takes to make a product, and I *KNOW* what can and cannot be done by a small group of entrepreneurs or volunteers. Accordingly, I would appreciate less ad hominem arguments as to what I understand and do not, and a bit more joining of these issues on the merits.

Unlike Craig, I do not presume that my background and contributions permit me to simply ASSERT, without more, what I believe to be the truth, expecting others to accept or simply withdraw. Of course, I proffer my advice and guidance (though no legal advice, of course :-)) on these issues, and will defend these points in the hope they might be helpful. Take them or leave them, but neither Craig nor his readers are diminished as people because they do not accept my views on faith. Challenge them fairly and directly, and I will probably give a fair and direct response within the reasonable limits of my time and patience. I believe Karsten will vouch for me on this point. I suggest that Craig adopt a similar view.

* the parade of horribles

Craig made several salient points here, and I do not address them all. A fair summary is this:

> I've not seen any evidence that
> > it has helped at all, and it *will* significantly hurt the entire
> > industry within ten years, *noticably*, as software itself makes
> > choices regarding what algorithms to apply to new problem domains
> > without the awareness of the humans posing the problems.

LPF made the same claim in 1991, but the parade of horribles never occurred. Why is that? Clearly, Craig's claim is presently non-falsifiable, but at least it bears some challenge on its face. I *DO NOT DOUBT* that some bad will result from the patent system as applied to software, and that war stories will be told. Particularly when viewed in the short term, some can be enormously harmful in the small. When viewed in the large, however, such stories will I believe be seen in the balance only one side of an overall cost and benefit to society.

Many "horror stories" serve in my mind as examples of the system working as it should. The Comptons multimedia patent played out exactly as it ought to, and should be a model for industry when threatened by bad patents. Reexamination is not a great device in practice for defendants in a specific action, but it *is* an excellent low-cost means to obtain credible review of af patents in the face of prior art on a shoestring budget. If the present patent reform bill is passed as originally proposed in this light, reexamination could open up substantially as a cost-effective alternative to litigation. This addresses the "bad software patents are bad" argument only, I will acknowledge.

Legitimate assertion of "good" software patents, it seems to me, is a reasonable result of the patent system. How to deal with it? Engineering around the patent is the most cost-effective solution; most software patents are far narrower than they appear, and work-arounds can often be done (often with the result that a new and better invention is developed -- a pleasant side-effect of the patent regime). Obtaining patronage and licenses, eithe from the patentee or others to respond to it and provoke cross-licensing is another. Finally, if OSS gets its act together, and is actually contributing to the body of knowledge, it can use its licenses and its own patents in defense.

It is the nature of IP, which must balance at least two inherently inconsistent interests: (1) the interest of inventors to benefit from the fruits of their invention; and (2) the interest of others (including inventors) to benefit from free and fair commerce and to stand on the shoulders of those who have come before them.

Accordingly, it is naive for supporters of any IP system to pretend anything is all good or all bad, but to acknowledge weaknesses inherent in the system. Nor do I pretend that minor repairs are not reasonable -- I *do*, if you will, think that repairs to a well-oiled and effectively operating machine, however inefficient, should be made carefully and with full appreciation of all of its implications. I believe proposals I have seen to date to be "knee-jerk" and ideological responses, indeed, responses designed to undermine rather than support the fundamental policies, but rather than deal in generalities and in platitudes would prefer to speak specifically and concretely on each in turn.

My prediction for the future is this: I don't think the sky is falling. I *do* think that the software patent system will continue and exdpand for decades, on balance more to the benefit than to the detriment of society. I *do* think that OSS-like efforts can in time benefit more than be harmed by the system, particularly if they "get with it," rather than ignore the way things operate.

This view is as non-falsifiable at present as is Craig's, and time will tell. In the meanwhile, however, I would make the following observations:

(1) Software patents and the patent system is largely misunderstood. Responsibly addressing what it is and is not benefits everyone. Proferring slogans and ideological lcok-step responses is not helpful, and often leads to bad results that could have been avoided.

[Craig's near-abandonment of g77 is a case-in-point. Had he been well-advised, it is unlikely he would have come close to such a

consideration. I would be interested in knowing precisely which patents were asserted against him that caused him to think about this.]

(2) The software patent system is not going away in fact. Indeed, State Street Bank has removed the last barriers (at least in public perception) as to questions whether a software invention faces Section 101 deficiencies. Accordingly, it is necessary for OSS movement to devote at least some of its efforts to finding ways to work with the patent system, both offensively and defensively. Public radio and television is an excellent model. Suggestions made by KarstenSelf and MySelf to obtain patrons and pro se attorneys is another avenue.

(3) The debate needs to get less acrimonious and more truth-seeking. It is unacceptable simply to repeat dogmatic lock-step conclusions. The answer clearly lies in the middle and that middle needs to be found, or we all rate to suffer as a society. Reversion to ad hominem attacks are both inappropriate and unfortunate -- they prove nothing and marginalize EVERYONE, not just the person so insulted.

YES, the patent system is not perfect, but
NO it is not the Great Satan either.

Software patents are a reality. Deal with it. Undertand them. Know what you are truly getting into, and deal with the consequences. You are not impotent, particularly if you are well-informed. It is not the case that software patents are inaccessible if you are working on a shoestring budget, particularly if you are working for the benefit of the public. There ARE things that can be done, and indeed, there ARE things that SHOULD be done. Gainsay is fine so far as it goes, but it doesn't answer any of the questions that need answering: more is required in a civilized debate.

And one aside, addressing more to the tenor of the debate than to their merits: attempts to marginalize arguments made by lawyers simply because they are made by lawyers "who get rich" by exploiting software patents are as offensive as they are trite and incorrect. Lawyers do not "get rich off" software patents, and there is PLENTY of work for lawyers, with or without software patents. I made far more money as a computer game designer than I will likely ever make as an attorney. Some of us disagree with Craig's positions, but our positions are not discredited merely for that reason. Nor is it meaningful to set up straw men and argue that MY position is discredited because Craig's straw man is indefensible. I would appreciate if Greg would address those arguments I made, rather than those I did not.

--
just another view,
Andy Greenberg
(remove "nonspam-" to get my real mailing address)

Subject: Re: Pamela Samuelson panelist at Open Source town meeting 8/21/98 San Jose, CA
Date: Wed, 12 Aug 1998 18:13:35 +0000
From: "Karsten M. Self" <kmself@ix.netcom.com>
Message-ID: <35D1DB4F.1001BC6A@ix.netcom.com>
References: <werdna-1208981010040001@tstpa2-11.gate.net>
Newsgroups: misc.int-property,misc.legal.computing,comp.os.linux.misc,gnu.misc.discuss
Lines: 94

I'll keep my comments brief.

Andrew C. Greenberg wrote:

> Craig feels I missed his point. I don't think so, but it is clear he has
> not responded to mine. I feel he has made several errors in his
> reasoning:
>
> * Free-software equates to "the entire software industry"
>
> Craig accurately observes that the traditional economic incentives
> provided to commercial businesses and individuals by the patent system do
> not accrue to the benefit of a volunteer who claims no interest in a free
> software product. This is undeniable. He then proceeds to observe that
> the existence the patent system nevertheless adds costs and expense to the
> process of developing all software, and hence also free software. This,
> too, is undeniable. By simple arithmetic, he concludes that there is a
> net loss under the patent system for free software.

I'll make enemies of both sides by saying that neither understands where OSS is going (I of course see the light). I feel that it will become a major part of the commercial IT industry, with major corporate players holding important roles. Yes, there will be room for small companies and volunteer efforts, but the fact that programmers need to by paid (by someone) necessitates some involvement of the business community.

As the business world recognizes the interest, and vulnerabilities, of OSS, I see changes to patent, but largely at first in how current law is utilized in specific licenses and corporate contracts. Andy recognizes this concept:

> This is misguided as well. Karsten and myself have both suggested ways in
> which low-rent, but effective use of the patent system can be obtained.
> Both involve obtaining the patronage and sponsorship of others, but that
> is, after all the point. If free software is a benefit to society (as
> many people believe public radio and television are a benefit), then it
> will make the effort to obtain such patronage. Just as volunteer effort,
> as well as housing and hardware are provided, so too can the patronage of
> industry and lawyers. If it can't, this informs the debate as to the
> social merits of free software.

Good point on social merits.

Changes in case law and statute would likely follow, and with a
significant lag. IMO, OSS cannot wait for the time it would take legal
mechanisms to work, and should exploit commercial interests to attain
the protection it seeks.

Much deleted. I agree with the general characterization of Craig's
comments. I think Craig is well intentioned if ignorant of the legal
aspects of what he speaks, much as I was here recently. I don't think
Andy would respond at such length and depth if he didn't see some merit
in doing so.

The followng are salient and worth repeating.

> This view is as non-falsifiable at present as is Craig's, and time will
> tell. In the meanwhile, however, I would make the following observations:
>
> (1) Software patents and the patent system is largely misunderstood.
<snip>
> (2) The software patent system is not going away in fact. Indeed, State
<snip>
> (3) The debate needs to get less acrimonious and more truth-seeking. It
<snip>
> Reversion to ad hominem attacks are both
> inappropriate and unfortunate -- they prove nothing and marginalize
> EVERYONE, not just the person so insulted.
>
> YES, the patent system is not perfect, but
> NO it is not the Great Satan either.
>
> Software patents are a reality. Deal with it. Undertand them. Know what
> you are truly getting into, and deal with the consequences.

<snip>

> Andy Greenberg

--
Karsten M. Self (kmself@ix.netcom.com)

What part of "gestalt" don't you understand?
Welchen Teil von "gestalt" verstehen Sie nicht?

web: http://www.netcom.com/~kmself
SAS/Linux: http://www.netcom.com/~kmself/SAS/SAS4Linux.html

10:51am up 60 days, 8:20, 3 users, load average: 1.40, 1.22, 1.03

Subject: Re: Pamela Samuelson panelist at Open Source town meeting 8/21/98 San Jose,
CA
Date: 13 Aug 1998 10:28:25 -0400
From: Craig Burley <burley@tweedledumb.cygnus.com>
Message-ID: <y667fxrm5y.fsf@tweedledumb.cygnus.com>
References: <6qfn8t$sem@bourbon.cs.umd.edu>
<35C74BAC.547245AC@ix.netcom.com>
Newsgroups:
misc.int-property,misc.legal.computing,comp.os.linux.misc,gnu.misc.discuss
Lines: 107

"Karsten M. Self" <kmself@ix.netcom.com> writes:

> I'll make enemies of both sides by saying that neither understands where
> OSS is going (I of course see the light). I feel that it will become a
> major part of the commercial IT industry, with major corporate players
> holding important roles. Yes, there will be room for small companies
> and volunteer efforts, but the fact that programmers need to by paid (by
> someone) necessitates some involvement of the business community.

Karsten, you can't come *close* to knowing whether I understand
where OSS is going, since you haven't learned what my views are
about it -- I haven't posted them at any length in *years*, IIRC.

If you'd like to call me and talk with me about this, I'd be more
than willing in your case, as you seem to have both some grasp,
and some genuine curiousity, plus a genuine desire and a real
ability to share what you have learned.

> Much deleted. I agree with the general characterization of Craig's
> comments. I think Craig is well intentioned if ignorant of the legal

> aspects of what he speaks, much as I was here recently. I don't think
> Andy would respond at such length and depth if he didn't see some merit
> in doing so.

I think Andy has mischaracterized my comments fairly consistently throughout, and I haven't seen much evidence that he's viewed this discussion as an attempt to educate me, so much as to ridicule me.

You've taken a different approach, thankfully, but I find your preferring Andy's follow-ups to my posts to my original posts rather worrisome.

As far as what you found worth repeating: "time will tell", indeed, but, years ago, when I debated these issues on USENET, software-patent attorneys (well, one in particular) seemed to not think free software would amount to much at all.

Well, that has changed, and, there is *no* evidence it was due to any benefits from software patents that I've seen posted, and there's *already* evidence free-software development has been *slowed down* by software patents.

Why I'm so "uninformed" by pointing out that this *suggests* that, indeed, the software patent system might be more expensive than is worthwhile, is beyond me. I've yet to see anyone actually debate that fundamental point, e.g. explain why free-software development will actually go *faster* and *cheaper* in the presence of software patents (admittedly a tough thing to do).

Oh, one lovable snippet from an old USENET conversation on the topic, by a fellow named Bart Lidofsky:

"The holders of the software patents are aware of the precarious ground on which they stand, and carefully calculate royalties so as to make it much less expensive to give in than to fight (patent cases are notoriously expensive; my father bought me a brand new car on the after-tax and expenses fee he got from a single day of testimony. The testimony? That the fact it takes 4 equations to solve for 4 variables is a law of mathematics)."

Finally, I've finally found something in my archives, in a post from Terry Ritter circa September 1991, following up one of my posts, in a thread discussing software patents. This snippet pertains to free software (again), though that wasn't the thrust of my involvement in the thread (again):

>>Again, software patents ultimately means no way for free software to
>>continue, at least not without massive support by the proprietary software
>>industry (not very likely).
>
>The problem is not patents.
>
>The problem is that even free software needs support and maintenance,
>but there is no cash flow for it. Support is important. Even when
>software is free, without support (and continued development) it's
>almost worthless.
>
>The problem is not "software patents." The problem is the lack of
>a business basis for support.

The reason I resurrect this is to illustrate the belief that existed among *strong* supporters of software patents that the system itself couldn't be at fault, but free software somehow was. (Also, I pat myself on the back a bit by posting my comment suggesting the need for the kind of system that Karsten proposes, though I did claim it wasn't very likely back then. I think it's more likely now.)

Now that there *is* and *has been* a business basis for support, there *still* exists a problem with software patents, which could conceivably shut down Linux, egcs, Apache, etc. *today* (well, pretty doggone quickly) with suitable litigation.

So it's nice that we're at least at the point where most people can agree that free software no longer has this "problem" of a "lack of a business basis for support".

Leaving the pertinent question as: "must this support be forced to also include defense against software-patent lawsuits, or is it more reasonable for society to conclude that software patents not be applicable to free software at all, thus saving all the expense and bother, given that all the pertinent inventions will be published anyway?"
--

"Practice random senselessness and act kind of beautiful."
James Craig Burley, Software Craftsperson burley@gnu.org

Subject: Re: Pamela Samuelson panelist at Open Source town meeting 8/21/98 San Jose, CA
Date: Thu, 13 Aug 1998 18:10:20 GMT
From: ritter@io.com (Terry Ritter)
Message-ID: <35d32c05.5936460@news.io.com>

References: <y667fxrm5y.fsf@tweedledumb.cygnus.com>
Newsgroups:
misc.int-property,misc.legal.computing,comp.os.linux.misc,gnu.misc.discuss
Lines: 97

On 13 Aug 1998 10:28:25 -0400, in
<y667fxrm5y.fsf@tweedledumb.cygnus.com>, in misc.int-property Craig
Burley <burley@tweedledumb.cygnus.com> wrote:

>[...]
>Finally, I've finally found something in my archives, in a post
>from Terry Ritter circa September 1991, following up one of my
>posts, in a thread discussing software patents. This snippet
>pertains to free software (again), though that wasn't the thrust
>of my involvement in the thread (again):
>
>>>Again, software patents ultimately means no way for free software to
>>>continue, at least not without massive support by the proprietary software
>>>industry (not very likely).
>>
>>The problem is not patents.
>>
>>The problem is that even free software needs support and maintenance,
>>but there is no cash flow for it. Support is important. Even when
>>software is free, without support (and continued development) it's
>>almost worthless.
>>
>>The problem is not "software patents." The problem is the lack of
>>a business basis for support.
>
>The reason I resurrect this is to illustrate the belief that
>existed among *strong* supporters of software patents that
>the system itself couldn't be at fault, but free software
>somehow was. (Also, I pat myself on the back a bit by posting
>my comment suggesting the need for the kind of system that
>Karsten proposes, though I did claim it wasn't very likely back
>then. I think it's more likely now.)
>
>Now that there *is* and *has been* a business basis for support,
>there *still* exists a problem with software patents, which
>could conceivably shut down Linux, egcs, Apache, etc. *today*
>(well, pretty doggone quickly) with suitable litigation.
>
>So it's nice that we're at least at the point where most people
>can agree that free software no longer has this "problem" of a
>"lack of a business basis for support".

It may come as no particular surprise to find that I, at least, do
not agree. I doubt that my opinion has changed much from my
published 1991 response to the LPF:

http://www.io.com/~ritter/ARTS/POLIPAT4.HTM

There are many businesses of which I would not be proud. One of these
is the business of supporting a product is provided without guarantee
of quality, even if the "guarantee" is simply a responsible vendor.
Can a free product have such guarantees? If so, who *really* pays to
make those guarantees good? Frankly, this sounds like a way to force
"the customer" into custom development, by *calling* it "support." I
don't know what this "support" is, but normal product support is paid
for by all users, and each support development occurs just once.

It seems to me that one of the biggest and most unfortunate
differences between some software businesses and hardware is the
general "hack" quality of software production. In my experience,
hardware production is test, test, test, test, test, while software
production is "no obvious bugs, so ship it."

I see quality is a contract between producer and user, delivery for a
price, independent of implementation technology. And when the
economic situation is set up so that contract *cannot* be enforced --
even in minor ways for heinous problems -- I think society needs a
different solution.

Things do not have to be this way. Both abstractly and in practice
software *is* hardware: When "software" runs, there is no software
there, just bits and gates and cells and voltages. Software does not
function; it describes and customizes a machine which functions. If
hardware can be built reliably, so can software. The missing
quantities here are the *will* to produce quality software and stand
behind it, and the *money* to make that happen. But it is very
difficult for quality to compete with free stuff.

We certainly have ample evidence that for-profit software production
does not necessarily produce high-quality software, and there is some
evidence that some free software can be good. Nevertheless, in free
products I see a fundamental disconnect between the producer and the
user which seems inappropriate to me.

What I would *really* like to see are public-domain OS interface
standards, implemented by multiple OS vendors, which would hopefully
provide the customer with a range of implementations of different
quality and price. Applications would then be competing on the same
playing field, with no hidden "OS tricks" or advance knowledge to

stack the deck.

Terry Ritter ritter@io.com http://www.io.com/~ritter/
Crypto Glossary 1998-08-12: http://www.io.com/~ritter/GLOSSARY.HTM

Subject: Re: Pamela Samuelson panelist at Open Source town meeting 8/21/98 San Jose, CA
Date: Fri, 14 Aug 1998 06:08:42 +0000
From: "Karsten M. Self" <kmself@ix.netcom.com>
Message-ID: <35D3D46A.B6F3C1B2@ix.netcom.com>
References: <6qfn8t$sem@bourbon.cs.umd.edu>
<35C74BAC.547245AC@ix.netcom.com>
Newsgroups:
misc.int-property,misc.legal.computing,comp.os.linux.misc,gnu.misc.discuss
Lines: 94

Craig Burley wrote:
>
> "Karsten M. Self" <kmself@ix.netcom.com> writes:
>
> > Andy prefers a minimal interventionist/minimal change route. This can
> > be a useful counterpoint. He's not completely averse to the effects of
> > substantive persuasion.
>
> It'd be good to see more evidence of that. Note that I, too, prefer
> that route. I wish the minimal route had been chosen, instead of

I didn't say that he's not argumentative as hell and as rough as can be on the clueless and trolls. My own history here started late May/early June. Try searching DejaNews for my email in mist.int-property. He and I were doing some big-time head-butting. Finally straighted out hard feelings in some offline email -- I said I'd pretend he wasn't as annoying as he seemed, and he agreed to pretend I'm not as stupid as I seem. Fact that I was willing to do some legwork, read up on law, cases, and articles helped.

BTW, you might want to pay a visit to http://www.findlaw.com/. You'll find the United States Code (U.S.C.), Supreme & Circuit court cases (to 1893 and various dates in the 1980s, respectively), state law and case studies, and scads of other stuff. Very valuable.

> just granting patentability to software with little or no public
> discussion.

AFIAK, this sort of emerged from a change in the courts' interpretation of the law, much as the origins of copyright grew from a change in the Copyright Office's

> > Me and my big mouth.
>
> *You* didn't start it, you didn't even follow up my first two entries
> into it, IIRC!

You bet your sweet ass I did!! I posted the original "Pam Sam at OS Town Mtg" message. No, I didn't respond to your original post. I had read the Forbes article -- actually it's what inspired the Patent Patron idea -- the discussion of what IBM could offer to Apache. I read that a couple of times and said, "wait a minute....". The rest is history. Picked up on you after your first dust-up with AG.

> > I'm not a lawyer. Quasi-informed on the law, and passionate about OSS.
> > I defer to Andy's knowledge of legal matters unless I can find
> > contradictory evidence.
>
> I'd like to see more posting of his personal legal and business
> experience showing how the industry couldn't survive (or flourish,
> at least) without software patents, and fewer mischaracterizations
> of what I've written.

He's a lawyer, you're the opposing counsel. If you want information damaging his position, you'll have to supply it. Depending on his mood, he may or may not conceed, he's not the judge either (sometimes he forgets this).

Note I've looked for arguments favoring patent on economic and societal grounds. Haven't found much. Note that it's usually the attackers of the status quo who are most vocal -- those who benefit by it are too busy reaping their rewards.

> I might as well leave the discussion, since I don't really have a
> problem "now" with software patents -- they're still practially
> not on my radar screen, they haven't significantly slowed down or
> stopped the free-software community yet, and when they do, there
> is a *wide* variety of possible responses, which, in a community
> like this, are likely to all be employed all at the same time
> by different people. :)

Well, if you do take off, I'd like to thank you for the Two Worlds suggestion. It's one of three viable defenses/alternatives I see for OSS and patents right now. Others are Patent Patron and Andy's Public Radio/ASPAC model.

> James Craig Burley, Software Craftsperson burley@gnu.org

--
Karsten M. Self (kmself@ix.netcom.com)

What part of "gestalt" don't you understand?
Welchen Teil von "gestalt" verstehen Sie nicht?

web: http://www.netcom.com/~kmself
SAS/Linux: http://www.netcom.com/~kmself/SAS/SAS4Linux.html

10:41pm up 61 days, 20:10, 2 users, load average: 1.09, 1.19, 1.19

Subject: Re: Pamela Samuelson panelist at Open Source town meeting 8/21/98 San Jose, CA
Date: Wed, 12 Aug 1998 10:48:13 -0400
From: werdna@nonspam-gate.net (Andrew C. Greenberg)
Message-ID: <werdna-1208981048130001@tstpa2-11.gate.net>
References: <y6hfzjl48l.fsf@tweedledumb.cygnus.com>
Newsgroups: misc.int-property,misc.legal.computing,comp.os.linux.misc,gnu.misc.discuss
Lines: 121

Having dealt generally with Craig's extensive posting with a regrettably lengthy posting of my own, I would like to address a few salient points that struck my interest:

I. Craig's "Two Worlds" Proposal

> > IMO, the solution is simple: pass laws that rule out *all*
> > software licensed under conditions that meet Open Source (TM)
> > (or similar) guidelines, such as the GPL, as targets of any
> > software-patent-related litigation.
> >
> > In other words, since an entire subclass of a functional industry
> > *always* publishes its inventions, don't allow it to be threatened
> > by a beauracracy that exists solely to ensure inventions are being
> > published.

This is a very interesting solution to the "problem" Craig raises. Presuming that Craig is correct about the costs and benefits, what would be the harms, if any, of such a regime? There exist compulsory licenses in other areas of IP protection, why not a CL for OS software (perhaps as a function of revenues generated from the software, which would be zero if the software is, in fact, free)?

By the way, I am ignorant as to what are the "GPL guidelines" and would appreciate a pointer. (or is that simply the Gnu Copyleft stuff?)

I see a few issues worth noting:

(1) How to avoid predatory practices by a company who wishes to undermine another commercial competitor by exploiting their patents in a free giveaway of an infringing product, thus at the same time neutralizing the competitor's market for the patented features? Would such laws effectively undermine the patent system at large by providing incentives for those at a competitive disadvantage to "explode" the patent by making OS software to that effect?

(2) The patent system requires an enabling disclosure. OS simply requires a manifestation of the invention. These are not the same thing. Moreover, what is the public benefit of the CL for OS software? It doesn't encourage the disclosure of new software inventions in OS, merely the disclosure of further exploitation of already disclosed inventions.

Finally, I am not sure how society benefits by providing that a person who has a legitimate patent covering an invention that can implemented in software benefits by letting others knock off the product and giving it away for free. If the original product's creation was beneficial to society, how does eliminating the incentive to create it and disclose its operation by subsidizing through a free compulsory license anyone who knocks it off, but also gives it away for free foster that benefit?

Craig appears to presume that the sole purose of the Patent clause and the Patent Act is to foster disclosure. He is mistaken in this regard (although disclosure is certainly a big art of the societal benefit). The purpose is to provide incentives with respect to "public goods" through limited monopolies. Monopolies limited with holes big enough to drive a truck through them provide no such incentives.

I do not wish to debate here the virtue of the monopoly so much as to point out that Craig is not "merely" proposing a parallel system that provides the same benefits in a different way. It appears instead that his proposal would provide no further devlopments, and would undercut the former. Nevertheless, the proposal is intersting, and perhaps a more refined version of it might address the complex competing policies we

would all wish to promote.

II. Nihilism

> In the long term, society gains no benefit from software patents as
> applied to free-software (or Open Source (TM)) products, so it is
> best that they should not be applicable to them at all, legally or
> otherwise.
>
> If the non-free developers have to be protected against free
> developers with software patents, then they have already lost
> the battle -- and perhaps their best option is to throw off
> the shackles of their *own* patent protection to better compete.
> (As a sometimes-proprietary developer, I think that would be best.
> I do have some reservations about whether society would need the patent
> system for software run as *services*, e.g. via Network Computers,
> which is thus never actually published, and is even more of a
> problem than software published only as difficult-to-reverse-engineer
> binaries in terms of sufficing.)

Craig here argues that "Patents are Bad," that in the absence of all IP protection, software will nevertheless be created, and people will continue to invent. I think history proved otherwise, showing that nations with strong IP have been substantially more inventive and substantially stronger in most meaningful areas of technology than those which have not.

The issue is not whether, after-the-invention-is-disclosed, a software company cannot compete with a generic knock-off artist who can on-the-cheap exploit that technology being unburdened with the cost of having developed the new technology (and the countless technologies that didn't work). The second product in the market is often much cheaper than the first one for precisely those reasons -- with the benefit of hindsight, a product can often be cheaper, better, faster and more responsive to market needs.

The issue is whether before-the-invention-is-invented, resources will be dedicated to making the invention -- or to making improvements on the invention. It is not whether software that is made by others for me to have for free is a good thing, but whether the software not yet invented would ever have been made.

Craig spends much energy distinguishing his g77 compiler from commercial compilers and the impact software patents had on his completion of his compiler. I am curious what software patents, if any, were so threatening as to meaningfully preclude its completion? Where any actually threatened? (I propose discussing the specific patents as examples of both his and our previous discussions). I was surprised to hear that there were issues, as most compiler technology, particularly for "traditional" programming languages, are very well-understood and particularly well-documented in the literature. Indeed, I didn't see how software patents implicated this at all -- and would be very interested to hear the details.

--
just another view,
Andy Greenberg
(remove "nonspam-" to get my real mailing address)

Subject: Re: Pamela Samuelson panelist at Open Source town meeting 8/21/98 San Jose, CA
Date: Wed, 12 Aug 1998 22:39:27 -0400
From: werdna@nonspam-gate.net (Andrew C. Greenberg)
Message-ID: <werdna-1208982239270001@tstpa2-80.gate.net>
References: <6qsj3r$1k01@bugsbunny.matrox.com>
<werdna-1208981048130001@tstpa2-11.gate.net>
Newsgroups:
misc.int-property,misc.legal.computing,comp.os.linux.misc,gnu.misc.discuss
Lines: 35

In article <6qsj3r$1k01@bugsbunny.matrox.com>, "Ryan Drake"
<rdrake@dont.spam.matrox.com> wrote:

> Just my two cents... I'm sure glad no one was able to patent "Calculus" or
> "Physics". If that were the case we'd probably still be driving around in
> horse-drawn carriages...
>
> Although I believe patents are a good way to protect specialized inventions
> (gadgets OR software), I think some processes do humanity more good when not
> patented.

The Supreme Court (and I) agree with Ryan.

It's very important to understand the distinction between patentable and unpatentable subject matter. While Calculus and Physics (as abstract laws and formulae) are expressly excluded from patentable subject matter, so too are mathematical algorithms, and in PRECISELY THE SAME WAY.

You cannot patent a formula, pure algorithm or law of nature, per se. However, you CAN patent the application of such a formula, algorithm or law of nature to a concrete and meaningful application, when manifest as a concrete process or apparatus.

Thus, a patent on a pendulum might be permissible, while the particular
mathematics and physics would have remained free for other applications.

And, with all due respect, its just as reasonable suggest that if
horse-drawn carriages weren't patentable, as well as technologies derived
therefrom and developed thereafter, we'd probably still be driving around
in horse-drawn cariages.

--
just another view,
Andy Greenberg
(remove "nonspam-" to get my real mailing address)

Subject: Re: Pamela Samuelson panelist at Open Source town meeting 8/21/98 San Jose,
CA
Date: Thu, 13 Aug 1998 08:28:33 GMT
From: phm@a2e.de (PILCH Hartmut)
Message-ID: <ExMCvM.2xp@a2e.de>
References: <werdna-1208982239270001@tstpa2-80.gate.net>
Newsgroups:
misc.int-property,misc.legal.computing,comp.os.linux.misc,gnu.misc.discuss
Lines: 45

werdna@nonspam-gate.net (Andrew C. Greenberg) writes:

>It's very important to understand the distinction between patentable and
>unpatentable subject matter. While Calculus and Physics (as abstract laws
>and formulae) are expressly excluded from patentable subject matter, so
>too are mathematical algorithms, and in PRECISELY THE SAME WAY.

>You cannot patent a formula, pure algorithm or law of nature, per se.
>However, you CAN patent the application of such a formula, algorithm or
>law of nature to a concrete and meaningful application, when manifest as a
>concrete process or apparatus.

>Thus, a patent on a pendulum might be permissible, while the particular
>mathematics and physics would have remained free for other applications.

This shows where the problem with software patents is: Software is not
necessarily a commodity. It can be a made into a commodity, but it can also
viewed as a piece of text written in a particular formal language. The FSF
never speaks about software "products" but only about "works" or "pieces"
for this reason.

Software patents are only applicable to "products", not to "works". Whatever
is placed under Open Source license terms is only a work (abstract system
of ideas) and can never become a product (concrete process or apparatus).

Thus when a company uses a software patent to keep a competitor abreast,
that competitor should, according to the spirit of patent law as described
by you, have the option to retaliate by making contributions to the free
software community.

Patents would have an impact on the Open Source work though. They would
make the license conditions such that, as soon as somebody built a commodity
around or on top of this work, patent fees could be charged. This would
mean that everything built on the OpenSource work would again have to be
OpenSource, which is very similar to what the GPL aims at.

Making knowledge public was after all the purpose of the patent system,
wasn't it?

--
Hartmut Pilch <phm@a2e.de>
a2e Language Services Pilch, Wang, Fujita & Co
better communication by virtue of language barriers
http://www.a2e.de/oas/, Tel +49891278960-8

Subject: Re: Pamela Samuelson panelist at Open Source town meeting 8/21/98 San Jose,
CA
Date: Thu, 13 Aug 1998 07:46:18 -0400
From: werdna@nonspam-gate.net (Andrew C. Greenberg)
Message-ID: <werdna-1308980746180001@tstpa2-80.gate.net>
References: <ExMCvM.2xp@a2e.de>
Newsgroups:
misc.int-property,misc.legal.computing,comp.os.linux.misc,gnu.misc.discuss
Lines: 73

In article <ExMCvM.2xp@a2e.de>, phm@a2e.de (PILCH Hartmut) wrote:

> This shows where the problem with software patents is: Software is not
> necessarily a commodity. It can be a made into a commodity, but it can also
> viewed as a piece of text written in a particular formal language. The FSF
> never speaks about software "products" but only about "works" or "pieces"
> for this reason.

That might be the problem if a claim were directed to a piece of text
written in a particular formal language. It is important to remember that
a patent's monopoly id defined by the claims of a patent, and not by the
general subject matter identified or discussed in the title, abstract or
specifiction. Although the specification relates to the cosntruction of a

claim, the claim's the thing.

Claims are typically directed to apparatus and methods, and not to books. To the extent that a claim were directed to text on paper, there remains the printed matter exception. Remember, a patented invention must be useful, in the sense of having utility. For protection of expressions of an idea, a different regime (copyright) is used.

> Thus when a company uses a software patent to keep a competitor abreast,
> that competitor should, according to the spirit of patent law as described
> by you, have the option to retaliate by making contributions to the free
> software community.

I have never seen a case where "retaliation" by disclosing a publicly disclosed idea, or a manifestation of it would be actionable. The difficulty comes when the disclosure of the embodiment of the idea is "made, used, sold or offered for sale." Any one of those things is an inringement. Now, if I engage in conduct, knowing that my conduct is likely to lead to infringement and it does, I may not be responsible for direct infringement, but I would be responsible for indirect infringement. "Retaliation by making contributions to the free software community," expecting that the software would be used, would very likely constitute contributory infringement or an inducement to infringe.

> Patents would have an impact on the Open Source work though. They would
> make the license conditions such that, as soon as somebody built a commodity
> around or on top of this work, patent fees could be charged. This would
> mean that everything built on the OpenSource work would again have to be
> OpenSource, which is very similar to what the GPL aims at.

This *is* interesting. PatentLeft (should we call it a "Latent"?) is something I hadn't considered -- using patents to protect the balance of what is in an OSS program. Under Copyright law, given access to OSS source code, a copyleft can be avoided by a simple process called a "clean room," whereby a specification of "ideas" is developed, passed to persons who didn't see the source for development, and evaluated by the specification team for conformity with the original, thereby making it possible to prove that the new program, though clearly derived from the original, was not "copied" in a sense that would make it subject to a claim of copyright infringement. Such not-so-reverse engineering (given that they begin with source code) is constitutionally protected and clearly fair use.

Patents on the software would preclude the use of particular program structures and get closer to the "ideas" of the program, so that a Latent license, like a CopyLeft would more strongly enforce the OSS nature of the software and its progeny.

> Making knowledge public was after all the purpose of the patent system,
> wasn't it?

That is certainly one of the princpal purposes of the patent system -- of course it is not the only one. Disclosure in a specification is undoubtedly the quid pro quo for the patent monopoly resulting from allowance of claims enabled by that disclosure.

--
just another view,
Andy Greenberg
(remove "nonspam-" to get my real mailing address)

Subject: Re: Pamela Samuelson panelist at Open Source town meeting 8/21/98 San Jose, CA
Date: Thu, 13 Aug 1998 08:38:14 -0700
From: Bruce Hayden <bhayden@ieee.org>
Message-ID: <35D30866.6BD17597@ieee.org>
References: <ExMCvM.2xp@a2e.de>
Newsgroups: misc.int-property,misc.legal.computing,comp.os.linux.misc,gnu.misc.discuss
Lines: 45

PILCH Hartmut wrote:
>
> This shows where the problem with software patents is: Software is not
> necessarily a commodity. It can be a made into a commodity, but it can also
> viewed as a piece of text written in a particular formal language. The FSF
> never speaks about software "products" but only about "works" or "pieces"
> for this reason.

Whether it is a commodity or not is irrelevant. Also, whether it can be viewed as a piece of text is irrelevant. Software has in essence at least two different aspects, its literary content and its functionality, and both are potentially protected by very different laws.

> Software patents are only applicable to "products", not to "works". Whatever
> is placed under Open Source license terms is only a work (abstract system
> of ideas) and can never become a product (concrete process or apparatus).

I load software into a general purpose computer, and it becomes a special purpose computer. That is usually a machine under the patent laws.
What you say in your Open Source license is irrelevant.

> Thus when a company uses a software patent to keep a competitor abreast,

> that competitor should, according to the spirit of patent law as described
> by you, have the option to retaliate by making contributions to the free
> software community.

Absolutely does not make sense. Please expound.

> Patents would have an impact on the Open Source work though. They would
> make the license conditions such that, as soon as somebody built a commodity
> around or on top of this work, patent fees could be charged. This would
> mean that everything built on the OpenSource work would again have to be
> OpenSource, which is very similar to what the GPL aims at.

Again, I think that you are confusing license provisions with patentability.
--

The preceding was not a legal opinion, and is not my employer's.
Original portions Copyright 1998 Bruce E. Hayden,all rights reserved
My work may be copied in whole or part, with proper attribution,
as long as the copying is not for commercial gain.

Bruce E. Hayden bhayden@acm.org
Phoenix, Arizona bhayden@copatlaw.com
***Currently use the following: bhayden@uswest.net

Subject: Re: Pamela Samuelson panelist at Open Source town meeting 8/21/98 San Jose, CA
Date: 13 Aug 1998 07:07:32 GMT
From: Stefaan.Eeckels@ecc.lu (Stefaan A Eeckels)
Message-ID: <6qu3bk$e22$1@justus.ecc.lu>
References: <6qt7ehdvo1@jaka.ece.uiuc.edu>
<werdna-1208981048130001@tstpa2-11.gate.net>
Newsgroups:
misc.int-property,misc.legal.computing,comp.os.linux.misc,gnu.misc.discuss
Lines: 48

In article <6qt7ehdvo1@jaka.ece.uiuc.edu>,
peltz@jaka.ece.uiuc.edu (Steve Peltz) writes:
> ALL of
> these are "clever", but all can be easily "invented" by someone working
> on the problem. I don't think something is worthy of being patented if,
> by sitting 10 programmers down to solve a similar problem, 5 of them
> will come up with that same technique.
But the records of patent offices are full of 'obvious' solutions.
Quite often the 'obviousness' is post-factum; humans have a tendency
to discover complex solutions before they discover the simple ones ;-)

One of the problems with history is that it is selective - some
facts have never been recorded. From a passion for inventions
weird and whacky, I can assure you that patents have been delivered
for the most obvious (and ridiculous :-) mechanical contraptions.

What hasn't emerged from the computer age are the 'inventors'
persons who'd wrack their brains to come up with inventions,
not simply to use them once in a product, but to patent them (or sell
them to big companies). At least, that what they hope to do :-)

I guess it's an attitude problem ;-) but seriously, putting too
much emphasis on the 'speech' aspects of programs doesn't help
either. After all, a programs is basically the stuff that adapts
a non-specific tool or machine to a particular problem. Mutatis
mutandis, this is similar to the process that adapts a block of
metal or wood to perform a particular function.

IMHO, the case for 'free software' or OSS is that the best method
to ensure that important software (such as an OS or a networking
stack) is secure and correct, is to publish the source code and
make it available for public scrutiny. I wouldn't put faith in
programs (including my own) that haven't been scrutinized by a
number of independent reviewers, and that I cannot examine myself
(if necessary). Microsoft will have to realize that you cannot
have a reliable OS unless it's OSS ;-)

Just my $0.02

--
Stefaan
--

PGP key available from PGP key servers (http://www.pgp.net/pgpnet/)
_____
Perfection is reached, not when there is no longer anything to add,
but when there is no longer anything to take away. -- Saint-Exupery

---

Subject: Re: Pamela Samuelson panelist at Open Source town meeting 8/21/98 San Jose, CA
Date: 13 Aug 1998 11:14:43 -0400
From: Craig Burley <burley@tweedledumb.cygnus.com>
Message-ID: <y64svgsyl8.fsf@tweedledumb.cygnus.com>
References: <6qfn8t$sem@bourbon.cs.umd.edu>
<35C74BAC.547245AC@ix.netcom.com>
Newsgroups:

misc.int-property,misc.legal.computing,comp.os.linux.misc,gnu.misc.discuss
Lines: 297

werdna@nonspam-gate.net (Andrew C. Greenberg) writes:

> I see a few issues worth noting:
>
> (1) How to avoid predatory practices by a company who wishes to undermine
> another commercial competitor by exploiting their patents in a free
> giveaway of an infringing product, thus at the same time neutralizing the
> competitor's market for the patented features? Would such laws
> effectively undermine the patent system at large by providing incentives
> for those at a competitive disadvantage to "explode" the patent by making
> OS software to that effect?

Very insightful! I think we "think" the same way, looking for
ways that people might exploit supposed "solutions" to (perceived)
problems.

Note that Microsoft has been accused of doing the above anyway, ignoring
the patent issue.

Anyway, I admit that about all the up-front thinking I did along
these lines was "well, there'd be pressure to release free-software
variations of everything your competitor patented", a big grin
spread on my face, and I decided to stop thinking about it, and
hope nobody else thought about it. :)

> (2) The patent system requires an enabling disclosure. OS simply requires
> a manifestation of the invention. These are not the same thing.
> Moreover, what is the public benefit of the CL for OS software? It
> doesn't encourage the disclosure of new software inventions in OS, merely
> the disclosure of further exploitation of already disclosed inventions.

Good points, and all I can suggest offhand (not being nearly up on the
topic enough to discuss much further) is that the benefit would be
heightened if the rule applied to any free-software publications that
occurred since *application* of the patent, which means the public
had the invention available to it before public disclosure via the
patent system.

> Finally, I am not sure how society benefits by providing that a person who
> has a legitimate patent covering an invention that can implemented in
> software benefits by letting others knock off the product and giving it
> away for free. If the original product's creation was beneficial to
> society, how does eliminating the incentive to create it and disclose its
> operation by subsidizing through a free compulsory license anyone who
> knocks it off, but also gives it away for free foster that benefit?

It's the same kind of trade-off thinking that justifies patents in
the first place. If the original product's creation was beneficial
to society, the market would reward that creation, right? (Is there
any *other* objective basis to assess value?) And, given that the
reward exists, why need the patent system, especially in a field
where publishing is often effectively automatic anyway?

My answers are the typical ones regarding the value that *society*
places on well-constructed published descriptions of inventions,
sufficient to give society a reason to hand over monopolies for
the inventions.

> Craig appears to presume that the sole purose of the Patent clause and the
> Patent Act is to foster disclosure. He is mistaken in this regard
> (although disclosure is certainly a big art of the societal benefit). The
> purpose is to provide incentives with respect to "public goods" through
> limited monopolies. Monopolies limited with holes big enough to drive a
> truck through them provide no such incentives.

I distinguish between goals/objectives and designs and implementations.

If we wanted to provide incentive to "public goods", we could do that
without using anything quite like the patent system. E.g. appoint a
board of Overseeing Scientists to judge which published papers most
benefitted society (through practical inventions, whatever), and
have society (government) pay the publishers, accordingly.

We tend to prefer market mechanisms based on property rights (and
nurturing responsibilities) here, so we picked a system that rested
somewhat (perhaps uncomfortably) on that basis.

Same thing for copyrights and trademarks: it's easy enough to do
mind-exercises separating the various threads of the law, substituting
other approaches, and seeing what might happen (at least it is for me).

With patents, the *goal* is publication. Rewarding invention is
trivially done through market action -- the problem is encouraging
publication. Justifications about the "public good" and improving
the progress in the sciences and arts all stem from this.

Without the *publication*, none of this good accrues. Limited
monopolies would often be *effectively* granted "by nature" (an
unshared invention, kept secret, is one others can't easily use),
and government could even enforce that by providing an entirely
secret patenting system if all it wanted to do was reward
inventors beyond what the market might do.

So, first and foremost, our patent system is about encouraging publication. The rest of it is, AFAICT, all about "well, we like market mechanisms based on granting property rights, even if we have to create the property out of thin air, so let's go with that".

And I'm all on board with that, but continue (for many years now) to seriously question whether it's worthwhile for software, *especially* in an environment where publication of software reigns (fairly) supreme.

Given that patent law already excludes some uses for research purposes, it is not unreasonable to wonder whether the unique nature of the free-software community -- unique not so much because it is free, but because it is software -- deserves a special, total exemption from patent-infringement liability.

Offhand, I don't see easy solutions to the problems you pose, especially the one that I would word as "I want to get the upper hand on my competitors' software-patent portfolio, so I'll release trivial free-software products just encompassing those patents", other than to make the exemptions operate for only those free-software products released before publication (approval) of the patents.

And, that'd not do much to reduce the problems I see with software patents slowing down development, though at least it'd protect the developer from land-mine-type patents (by having him spend 90% of his volunteer time reading patents, or depend on others to do it for him -- others who *could* be using their time working for their big company, e.g. IBM, testing the software, etc.).

> I do not wish to debate here the virtue of the monopoly so much as to
> point out that Craig is not "merely" proposing a parallel system that
> provides the same benefits in a different way. It appears instead that
> his proposal would provide no further devlopments, and would undercut the
> former. Nevertheless, the proposal is intersting, and perhaps a more
> refined version of it might address the complex competing policies we
> would all wish to promote.

IMO I think it needs lots more refinement than that -- perhaps just tossing it out would be best. The only efficient alternative becomes eliminating software patents entirely.

Then again, patents are useful only when implemented in a useful way, so perhaps it isn't a feasible tactic to release trivial free-software programs to "attack" a competitors' software patents, as long as the patents still hold for non-free software (which must be the issue anyway, else why the attack?).

> Craig here argues that "Patents are Bad," that in the absence of all IP
> protection, software will nevertheless be created, and people will
> continue to invent. I think history proved otherwise, showing that
> nations with strong IP have been substantially more inventive and
> substantially stronger in most meaningful areas of technology than those
> which have not.

Or that nations that are substantially stronger in technology also end up with substantially stronger, more intricate laws, and smarter lawyers. :)

Clearly software gets created without software patents, because I've never seen a case where software patents have been even remotely involved in the authoring of software in *any* important way.

Since 1991, when I was debating this issue with those who felt software patents were wonderful and we couldn't get along without them, I've heard of *no* cases where software patents enabled the funding of free software, or even non-free software, but several cases of them making *new* (versions of) free software more difficult to develop have since occurred. (E.g. bzip2.)

> The issue is whether before-the-invention-is-invented, resources will be
> dedicated to making the invention -- or to making improvements on the
> invention. It is not whether software that is made by others for me to
> have for free is a good thing, but whether the software not yet invented
> would ever have been made.

That really cannot be the issue with software in general, unless it happens in areas of the industry I've never remotely encountered, and happens among people who can *predict* which patents will and won't be granted.

Admittedly, the myth of the "rare software patent" posted in favor of them around 1991 seems to have been just that, but...

...do you actually know *anybody* who has funded *true* R&D based on the granting, and subsequent planned receipts of license fees, of software patents? How many? How did they know whether their patent applications would be approved, and in how timely a fashion, or did they just scattershot and figure N% would be approved, relying on the threat of litigation to cover enough in fees later on to bring in the ROI?

Compare that to how many cases you know of where people have *lost*

time, money, and businesses due to being on the wrong end of
infringement lawsuits.

It just seems too much like a lottery for people to depend upon to
fund R&D, but obviously, having spun the wheel with excess
cash and gotten "lucky", more than a few people have been willing
to use the winnings to *eliminate* competitive R&D, or at least
competitive products, from the market -- including attacking
free-software products like X. (Just like, on the trademark side,
some guy "spun the wheel" on the Linux name, tried to profit off of
it at the expense of the free-software community, and lost -- but
not nearly as much as he should have, given that he basically
committed fraud, as I understand it, on the application by
representing the name as his own. Take enough risks like that,
with no real downside except an application fee here and there,
and You Could Be A Winner...or so might too many people think,
once free software becomes "de riguer" in some areas.)

> Craig spends much energy distinguishing his g77 compiler from commercial
> compilers and the impact software patents had on his completion of his
> compiler. I am curious what software patents, if any, were so threatening
> as to meaningfully preclude its completion? Where any actually
> threatened? (I propose discussing the specific patents as examples of
> both his and our previous discussions). I was surprised to hear that
> there were issues, as most compiler technology, particularly for
> "traditional" programming languages, are very well-understood and
> particularly well-documented in the literature. Indeed, I didn't see how
> software patents implicated this at all -- and would be very interested to
> hear the details.

At the time, it was clear the PTO hadn't a clue about existing
standards and practices in the software industry. My concern was
that I'd be sued over the approach I took to, say, resolve lexical
ambiguities, or store information on symbols. Register coloring
was an example that wouldn't have applied to me, since I wasn't
undertaking that part of the compiler -- but those who did, *did*
worry about it, IIRC. (I think that patent has since expired,
though.)

My views have not changed on software patents over the past 8 or so
years. Further, my *information* on how they're actually "going"
hasn't changed much, because I haven't paid much attention -- they're
not really on the "radar" much these days, especially since the
demise of the LPF, though how that might be related I can't say.

So I don't know that the threat is greater or lesser, other than
to say I think free software has come a long way in terms of being
a target in the past 8 years, and have heard of only a few recent
cases of software patents doing bad things.

But, my point was: free software shows that developing software
need *not* be expensive enough to define software patents as an
"affordable" expense in the process.

And, during the past 8 years, I've yet to see *one* post or claim
that a software patent application was part of up-front funding
to fund free-software development. It's tempting to conclude
that free-software developers are just clueless, but so many of
them, like myself, just do (as a "hobby") what they do full-time
(in my case, on occasional contract, or part-time) for proprietary
developers.

So, since *I* haven't observed any such up-front funding of
patentable materials to fund R&D, and since no free-software
people seem to be using that tactic, despite (as either you or
Karsten pointed out) the example of the GPL using copyright
to get around copyright law...

...it seems reasonable to conclude that the predicted *positive*
effects of software patents on software development have not
come about. Otherwise we'd have seen all sorts of success stories,
right?

So, since 8 years ago, I've seen only a few negatives, no
positives, no *evidence* of positives -- other than claims just
like the ones made 8 years ago -- not nearly enough to change,
or sigificantly solidify, my views on software patents.

What *has* changed a bit is the likelihood of NC-type computing
predominating, which is why I think there might be a legitimate
need for software patents in the near future, at least to cover
some areas. But it doesn't seem likely it'll be *enough* of a
need to justify the whole system applying to software, offhand,
though if people ever start posting *real* examples of how
up-front funding of patent applications enabled much larger up-
front funding of R&D into *software* techniques that otherwise
wouldn't likely have been gotten ahold of, then maybe minds like
mine will change.

(I happen to think that NC-type computing offers fairly substantial
disadvantages, and free-software-based computing similarly
substantial positives, on the technical side, so I'm not really
expecting things to go in the just-keep-unpublished-software-on-
our-server direction anytime soon enough to make preserving
software patents appropriate on that basis. But, I could be wrong,

and it'd be silly to ignore that possibility just to be able to
claim software patents could never be beneficial -- in theory,
if we had fulfilled Kerrey's [sic?] original vision, and each
had a *terminal*, but not a computer, in our homes, connected
to one of several huge servers, software patents might be the
only way to encourage publication of the pertinent inventions.
Instead, we have bazillions of PCs, and an increasing body of
source code.)

In the meantime, as I've said, all I've really seen are negatives,
and no positives.
--

"Practice random senselessness and act kind of beautiful."
James Craig Burley, Software Craftsperson burley@gnu.org

Subject: Re: Pamela Samuelson panelist at Open Source town meeting 8/21/98 San Jose,
CA
Date: Thu, 13 Aug 1998 08:24:44 -0700
From: Bruce Hayden <bhayden@ieee.org>
Message-ID: <35D3053C.99576407@ieee.org>
References: <m3soj1oh67.fsf@kiteless.dyn.ml.org>
  <6qsj3r$1k01@bugsbunny.matrox.com>
  <werdna-12089810481300001@tstpa2-11.gate.net>
Newsgroups:
misc.int-property,misc.legal.computing,comp.os.linux.misc,gnu.misc.discuss
Lines: 21

Decklin Foster wrote:
>
> "Ryan Drake" <rdrake@dont.spam.matrox.com> writes:
>
> > Just my two cents... I'm sure glad no one was able to patent "Calculus" or
> > "Physics". If that were the case we'd probably still be driving around in
> > horse-drawn carriages...
>
> If either were invented today, they could be.

This is of course incorrect.
--

The preceding was not a legal opinion, and is not my employer's.
Original portions Copyright 1998 Bruce E. Hayden,all rights reserved
My work may be copied in whole or part, with proper attribution,
as long as the copying is not for commercial gain.

Bruce E. Hayden bhayden@acm.org
Phoenix, Arizona bhayden@copatlaw.com
***Currently use the following: bhayden@uswest.net

Subject: Re: Pamela Samuelson panelist at Open Source town meeting 8/21/98 San Jose,
CA
Date: Thu, 13 Aug 1998 08:33:15 -0700
From: Bruce Hayden <bhayden@ieee.org>
Message-ID: <35D3073B.E2FD39AB@ieee.org>
References: <6qt7ehdvo1@jaka.ece.uiuc.edu>
  <werdna-12089810481300001@tstpa2-11.gate.net>
Newsgroups:
misc.int-property,misc.legal.computing,comp.os.linux.misc,gnu.misc.discuss
Lines: 54

Steve Peltz wrote:
>
> There are quite a few patents on various optimization techniques - one that
> I've heard of is "register coloring" (I have no idea what it actually is).

My memory is that this is an extension of the 3/4 color map problem.
Colors are used in register allocation. However, I don't remember for
sure. I got involved a couple of years ago writing some compiler patents,
and we considered this to be prior art at the time.

> I'll give a specific example of a related problem that I have no idea
> of what is covered by patents - a combined emulator/translator, such
> as DECs x86 emulator for their Alpha machines; given the problem of
> efficiently executing code for another processor, there are quite a few
> straightforward solutions. Straight emulation is, I would hope, totally
> free of liability (but, it could have been patented relatively recently,
> as machines fast enough to make emulation a satisfactory solution
> have only been affordable within the last 10-15 years; similar to the
> backing-store patent, although the idea may have been obvious, a patent
> may still have been granted since no one had ever bothered DOING it, since
> it wasn't practical due to the existing supporting technology). Statically
> translating code has been done (we used a static translator to take
> 8080 source code and assemble it for an 8086). Translating without the
> source is more difficult, because often data and instructions are mingled
> together, but there are several techniques, such as dynamic tracing of
> instructions. Dynamic translation (translating on-the-fly and caching the
> results for re-use) is another technique. There are all sorts of clever
> little things you can do (such as, don't translate a block until it has
> been executed at least a certain number of times, as straight emulation is
> faster than translation/execution for a single pass). There are register
> optimization techniques (especially if the emulated machine has too many

> registers to keep in registers on the implementing architecture). ALL of
> these are "clever", but all can be easily "invented" by someone working
> on the problem. I don't think something is worthy of being patented if,
> by sitting 10 programmers down to solve a similar problem, 5 of them
> will come up with that same technique.

But as Andy points out, definition of the problem is often much of
the invention. One of the things that you have to fight with
examiners all the time over is their human tendency to look at
your problem definition, and from that jump to your conclusion.
This is not allowed, if the problem definition is itself novel
and nonobviousness. And note that this problem arises across all
technologies, and is not limited to software.
--

The preceding was not a legal opinion, and is not my employer's.
Original portions Copyright 1998 Bruce E. Hayden,all rights reserved
My work may be copied in whole or part, with proper attribution,
as long as the copying is not for commercial gain.

Bruce E. Hayden bhayden@acm.org
Phoenix, Arizona bhayden@copatlaw.com
***Currently use the following: bhayden@uswest.net

Subject: Re: Pamela Samuelson panelist at Open Source town meeting 8/21/98 San Jose,
CA
Date: 13 Aug 1998 14:03:12 -0400
From: Craig Burley <burley@tweedledumb.cygnus.com>
Message-ID: <y6ogtosqsf.fsf@tweedledumb.cygnus.com>
References: <6qfn8t$sem@bourbon.cs.umd.edu>
<35C74BAC.547245AC@ix.netcom.com>
Newsgroups:
misc.int-property,misc.legal.computing,comp.os.linux.misc,gnu.misc.discuss
Lines: 39

Bruce Hayden <bhayden@ieee.org> writes:

> Ryan Drake wrote:
> >
> > Just my two cents... I'm sure glad no one was able to patent "Calculus" or
> > "Physics". If that were the case we'd probably still be driving around in
> > horse-drawn carriages...
>
> First of all, even under today's patent laws, either would be nonstatutory.
> More important though, even if they were, the stuff would have entered
> the public domain centuries ago.
>
> Indeed, with the history of Calculus, at least, given the spread of
> knowledge at that time, I cannot see that even if it had been patented
> (which of course it couldn't be), that any temporary monopoly would have
> had nearly zero impact.

Even more pertinent, I've read (in debates about software patents!)
that Newton actually *did* keep his Calculus secret for years, using
it for his own purposes, until he realized somebody else was about
to publish similar stuff, which triggered him releasing his, apparently
to get more credit.

Needless to say, this is an argument in *favor* of patenting entire
fields of mathematics, or at least algorithms.

Which also makes it a point in favor of recognizing that just because
some useful inventions might be "lost" without patent protection
in a field does not mean patent protection for that field is thus
entirely justified.

Otherwise we might as well allow *everything* to be subject to patent
law, since it's easy enough to come up with an example of something
that might have been published (or brought to market) faster with
the promise of a monopoly.
--

"Practice random senselessness and act kind of beautiful."
James Craig Burley, Software Craftsperson burley@gnu.org

Subject: Re: Pamela Samuelson panelist at Open Source town meeting 8/21/98 San Jose,
CA
Date: Fri, 21 Aug 1998 08:45:28 +0000
From: "Karsten M. Self" <kmself@ix.netcom.com>
Message-ID: <35DD33A8.2076DEE6@ix.netcom.com>
References: <Exzz6w.1zy@a2e.de>
<35D9D5EF.1BEAC6D4@ix.netcom.com>
Newsgroups:
misc.int-property,misc.legal.computing,comp.os.linux.misc,gnu.misc.discuss
Lines: 152

PILCH Hartmut wrote:
>
> "Karsten M. Self" <kmself@ix.netcom.com> writes:

> Is there a chance of getting rid of software patents altogether?

"Snowball's chance in hell" is the vernakular expression here in the States. Trend in IP for the 90s has been increasing, not decreasing, owner-interest for both copyright and patent law. You'd be bucking the trend. Andy Greenberg makes some very convincing arguments that even attempting to liberalize IP via legislative reform would probably have the reverse effect due to the workings of the political process -- the folks who have an interest in stronger IP protections will warp legislation before it's enacted.

> >Additionally, Two Worlds is potentially controversial due to the fact
> >that it asserts an entitlement to patented IP. This would likely not
>
> I don't understand this. IP is patented all the time. What is new about
> asserting such an entitlement?
>
> If I understand correctly, "Two Worlds" refers to a world of freely
> circulating information, including OSS, vs the material world, in which such
> information can be embodied, e.g. as hardware or that kind of quasi hardware
> which is produced by compiling, shrink-wrapping, removing source code and
> structure information and other crippling mechanisms designed to turn
> information into merchandisable "software".

I don't think we're on the same wavelength. Craig Burley suggested "Two Worlds of IP protection", one applying to OSS/free software, one applying to proprietary products. Given the justification of patent and copyright in the US Constitution, which says (paraphrased) that to promote the arts and technology, Congress can grant artists and inventors exclusive control over thier works, Craig's interpretation (which many agree with) is that patents exist to serve a greater social good.

He suggests that as OSS *is* open and reveals its methods, it ought to have free access to patents, without payment of royalties. This is what I would call an entitlement -- a mandated right to something. The problem is that this essentially says to the person who is providing the thing being entitled "you have no say in whether you want to grant this or not, you have to give it because the law says so". This leads to discontent. Not good.

I prefer a system in which the owners of patents can choose to share or not to share. I'd like to create a system which creates very strong incentives to share. But I'd like to leave the option of not sharing available, however costly and/or income-forgoing it might be.

> >In the US, music licenses are largely controlled by an organization
> >called ASCAP (http://www.ascap.com/). This is a non-govenmental
>
> In Germany, this is GEMA, in Japan Chosakken Kyoukai, I don't know the URLs.
>
> Unfortunately this organisation, just like the people in the patent system,
> is only interested in enlarging the scope of its claims. It would surprise
> me to find anyone there with any understanding of the problems IP creates
> for the rest of the world.

This is where business organization becomes critical. You want an what keeps them in existance. The trick is to make the things which benefit the organization be the same things which benefit its members, original intent, and stated objectives. It's not an easy thing, and I can already think of a couple of pitfalls the organizational approach would have to work carefully to avoid. This is a real problem, but it exists for any organization.

> I can't imagine at the moment how this can work. Why should patent holders
> agree to join some voluntary cooperative to surrender their rights?

Good question. Because:

- They have an interest in OSS as a business opportunity, and have the same risk as OSS developers and vendors do that an OSS product might be killed by a patent infringement claim. Even if they don't own the OSS code, losing a $1 billion market is losing a $1 billion market. If the billion.

- Because they are selling products or services based on OSS products and face the prospect of patent infringement liability themselves. Or worse, they're creating a liability for their customers. Patent infringement doesn't just apply to programmers -- vendors and end-users are liable as well. This is why the off-shore development suggestion doesn't work -- even if a program is written outside the US, if it infringes a US patent, the US vendors and users face risk of an infringement case.

- Because the organization offers access to other patents at a lower cost, and with less hassle, than would be available independently (members get a discounted rate for non-OSS use of patents).

- Because by pooling its patents with the organization, and making them available to OSS, the chance that the OSS community sees the patent as something to be invented around, and obsoleted, is reduced.

- Because the organization would also be involved in marketing and collecting royalties on the patent -- reducing administrative overhead for the patent holder.

- Because the organization would protect and defend the patent -- pursuing infringing use, and defending validity claims, reducing administrative overhead and legal costs.

> >web: http://www.netcom.com/~kmself
> >SAS/Linux: http://www.netcom.com/~kmself/SAS/SAS4Linux.html
>
> Any URLs on your projects or alternative solutions to the SW patent problem?

No, you're pretty much looking at it here in the newsgroups. The idea only really hatched in the last week, thought it's been festering for a little over a month. I've forwarded the suggestion formally to several of the powers that be, such as they are, in the Linux community -- ESR, RMS, Linux International, and am waiting to hear from them at this point. If you have any pull or sway with them and like this idea (or you hate it), I'd appreciate the plug (goes for anyone reading this, hint).

I'm working on what the organizational structure should be like, what the organization would do, and how to answer questions like the one's you're posing here.
I do want to put some thoughts down on a website somewhere, don't believe I'll have much time for it for the next week or so.

> --
> Hartmut Pilch <phm@a2e.de>
> a2e Language Services Pilch, Wang, Fujita & Co
> better communication by virtue of language barriers
> http://www.a2e.de/oas/, Tel +49891278960-8

--
Karsten M. Self (kmself@ix.netcom.com)

What part of "gestalt" don't you understand?
Welchen Teil von "gestalt" verstehen Sie nicht?

web: http://www.netcom.com/~kmself
SAS/Linux: http://www.netcom.com/~kmself/SAS/SAS4Linux.html

1:11am up 68 days, 22:40, 2 users, load average: 1.38, 1.71, 1.60

Subject: Re: Pamela Samuelson panelist at Open Source town meeting 8/21/98 San Jose, CA
Date: 12 Aug 1998 17:36:00 -0500
From: peltz@jaka.ece.uiuc.edu (Steve Peltz)
Message-ID: <6qt5cg$d10$1@jaka.ece.uiuc.edu>
References: <werdna-1208980844430001@tstpa2-11.gate.net>
<6qq9c4$84n$1@jaka.ece.uiuc.edu>
<y6hfzjl48l.fsf@tweedledumb.cygnus.com>
Newsgroups:
misc.int-property,misc.legal.computing,comp.os.linux.misc,gnu.misc.discuss
Lines: 38

In article <werdna-1208980844430001@tstpa2-11.gate.net>,
Andrew C. Greenberg <werdna@nonspam-gate.net> wrote:
>> And the Bingo patent isn't dead, another company was sued recently.
>
>I would be pleased to know more about this patent.

I believe it is 4,378,940, but I can't be positive. Where can I look at the claims on-line (USPTO only allows looking at the abstract).

While looking for this, I found some pretty odd ones, in particular 5,727,786 and related patents. From the abstracts, these look like methods of playing a game. I wasn't aware you could patent the rules of a game (such as a variation of poker, or in this case a variation on the rules of bingo).

Other bingo-related patents that look bogus (again, just based on the abstract - there could well be SOME claims I wouldn't think are bogus) are 5,718,631, 4,624,462, 4,848,771, 4,475,157, 4,455,025.

The problem with a lot of software patents, as I see it, is that the solution is obvious once you decide to solve the problem. For instance, if someone wanted an aid to playing bingo, so they could enter a bunch of cards into their computer and have it check each called number against all the cards and have an alarm indicate if any of the cards had a bingo, then indicate which card had the bingo and read-back the called numbers that make up the bingo - well, stating the need was 90% of determining the entire problem, knowing the game of bingo and what information might be needed to help you play it is another 10%, and any idiot can write the program that does that. Yet, that appears to be the patentable material in more than one of the above cited patents.

Given the basic game of bingo, it appears to me that you can not write a computer implementation of that without violating a dozen or more patents. YET, I have proof that such a game had been written, and was being played, on a publically accessed computer system in 1976, and such proof has been offered in defense against one patent (and was, at least initially, rejected by a judge who apparently didn't understand the concept of a time-sharing system vs. a distributed system).

Subject: Re: Pamela Samuelson panelist at Open Source town meeting 8/21/98 San Jose, CA
Date: Wed, 12 Aug 1998 23:08:23 -0400
From: werdna@nonspam-gate.net (Andrew C. Greenberg)
Message-ID: <werdna-1208982308230001@tstpa2-80.gate.net>
References: <6qt5cg$d10$1@jaka.ece.uiuc.edu>
Newsgroups: misc.int-property,misc.legal.computing,comp.os.linux.misc,gnu.misc.discuss
Lines: 61

In article <6qt5cg$d10$1@jaka.ece.uiuc.edu>, peltz@jaka.ece.uiuc.edu (Steve Peltz) wrote:

*snip stuff on Bingo. Thanks, Steve -- I'll educate myself and get back.

> The problem with a lot of software patents, as I see it, is that the
> solution is obvious once you decide to solve the problem.

Right. Deciding to define and solve a problem can in and of itself be sufficient to make the solution unobvious. You are not permitted to Monday Morning quarterback an invention, that is to use hindsight, in determining obviousness.

You construct the bizarre thing -- a person of ordinary skill in the art --, and impart upon that person knowledge of literally every published work in the art, and every published work in "analogous" arts (clearly defining the art is a big deal in determining obviousness). You impart upon that person the knowledge of every public use of anything in the art and anything publicly known in the art, and a whole bunch of other stuff. (You even impart upon the person all secret knowledge the inventor was given by others prior to the invention and everything secret that other persons would have known prior to the invention if they also invented the invention, but beforehand!)

Then, if that person could combine his knowledge to get the invention, without using the specification (including the problem definition), you've got him. But you can't combine references without some teaching in the reference suggesting the combination [remember, he's just an ordinarilly skilled hypothetical person.]

I described it this casual way to try to impart some of the sense in which legal obviousness differs from the English uses of the word. But be assured, if it was novel to apply software to the problem, that observation by itself can render the invention unobvious.

> well, stating the need was 90% of determining
> the entire problem, knowing the game of bingo and what information might
> be needed to help you play it is another 10%, and any idiot can write the
> program that does that.

I'll get back to you on that as well. But stating the need might itself have been an important observation. Many great, essential and elegant inventions were precisely that, asking the right question, the answer to which was obvious once the question was properly put.

On the other hand, this is not a hard-and-fast rule (what is in the law). After the transistor was disclosed, it wasn't too hard to observe that the thing is just a solid-state-relay, and for a VERY brief while, substituting a transistor for a relay in a particular circuit might have been patentable in its own right. But after a VERY brief time (in the case of the transistor) the substitution becomes an obvious alternative, and hence, merely noting the substitution isn't sufficient to avoid a 103 rejection.

I know its not what you wanted to hear, but I hope it helps you to understand some of what is going on.

--
just another view,
Andy Greenberg
(remove "nonspam-" to get my real mailing address)

Subject: Re: Pamela Samuelson panelist at Open Source town meeting 8/21/98 San Jose, CA
Date: 13 Aug 1998 16:23:03 -0700
From: jbuck@best.com (Joe Buck)
Message-ID: <6qvsgn$7pl$1@shell17.ba.best.com>
References: <werdna-1208982308230001@tstpa2-80.gate.net>
Newsgroups: misc.int-property,misc.legal.computing,comp.os.linux.misc,gnu.misc.discuss
Lines: 48

Andrew C. Greenberg <werdna@nonspam-gate.net> wrote:
>> The problem with a lot of software patents, as I see it, is that the

>> solution is obvious once you decide to solve the problem.
>
>Right. Deciding to define and solve a problem can in and of itself be
>sufficient to make the solution unobvious.

I am puzzled by this reasoning. I have decided to solve the problem
of playing Bingo on a computer. How can that decision alone make the
solution unobvious?

>You are not permitted to
>Monday Morning quarterback an invention, that is to use hindsight, in
>determining obviousness.

I am aware that this is how patent law works. But this principle runs
counter to the justification for patents: that the inventor trades
publishing of the patent for the limited monopoly, and thereby everyone
wins. If the publication of the patent is of no value to anyone (because
any skilled practitioner assigned the problem would come up with the
same solution, and because formulating the problem is of no value either,
because it is simply "automate some existing task"), the moral justification
for the patent (the one in the US Constitution) doesn't apply.

There is a patent for updating the cells of a spreadsheet by doing a
topological sort of the data dependencies. However, this is the only
correct way of solving the problem, and anyone with a computer science
education would solve the problem this way. Thus again, the publication
of the patent is of no value, as it teaches nothing. The consequence of
the patent is that spreadsheets are slower and cost more than they otherwise
would. The public is harmed by this patent.

>You construct the bizarre thing -- a person of ordinary skill in the art
>--, and impart upon that person knowledge of literally every published
>work in the art, and every published work in "analogous" arts (clearly
>defining the art is a big deal in determining obviousness).

Only in theory. In practice, the patent examiner searches the list of
existing patents, and isn't familiar with the literature. S/he sits
on the thing for a while, does a keyword search, sends back letters asking
for explanations of why the patent isn't the same thing as a dozen
irrelevant patents, and eventually grants it or doesn't.

--
-- Joe Buck
   work: jbuck@synopsys.com, otherwise jbuck@welsh-buck.org or jbuck@best.com
http://www.welsh-buck.org/

Subject: Re: Pamela Samuelson panelist at Open Source town meeting 8/21/98 San Jose,
CA
Date: 21 Aug 1998 01:40:02 -0700
From: tzs@halcyon.com (Tim Smith)
Message-ID: <6rjbp2$118$1@52-a-usw.rb1.blv.nwnexus.net>
References: <werdna-1508980015430001@tstpa1-34.gate.net>
   <6r2kt1$s05$1@halcyon.com>
   <werdna-1208982308230001@tstpa2-80.gate.net>
Newsgroups:
misc.int-property,misc.legal.computing,comp.os.linux.misc,gnu.misc.discuss
Lines: 43

Andrew C. Greenberg <werdna@nonspam-gate.net> wrote:
>What you do is to construct that hypothetical PHOSITA (person having
>ordinary skill in the art), endow him with the statutory knowledge, and
>ask whether it would be obvious to him to combine the preexisting art
>(absent the problem statement) to get to the patented invention. That's
>what you do. If you don't do that, you aren't talking about obviousness
>as the term is used in patentability.

Do you have a cite for that formulation of the test? That doesn't quite
match any test I recall from Professor Chisum's patent course, but I took
that a few years ago, and have not kept up with subsequent cases.

For concrete discussion, here are software patents that seem obvious
to me.

        5661665
        5659336
        5657050
        5649222

Those are assigned to Microsoft. I don't want to pick on Microsoft,
so here is one that just issued that isn't one of theirs:

        5790793

Here is claim 1 of that one:

1. A method of communicating between computers, comprising the steps of:

        creating a message at a first computer, said message including
        a reference to a predetermined location;

        transmitting, by the first computer, said message to a second
        location; and

receiving said message by a computer at the second location;

decoding said message by the computer at the second location
by retrieving data from the predetermined location,
automatically by a single application, without requiring user
interaction, into the computer at the second location.

--Tim Smith

Subject: Re: Pamela Samuelson panelist at Open Source town meeting 8/21/98 San Jose,
CA
Date: Fri, 21 Aug 1998 23:44:44 -0400
From: werdna@nonspam-gate.net (Andrew C. Greenberg)
Message-ID: <werdna-2108982344440001@tstpa1-71.gate.net>
References: <6rjbp2$118$1@52-a-usw.rb1.blv.nwnexus.net>
Newsgroups:
misc.int-property,misc.legal.computing,comp.os.linux.misc,gnu.misc.discuss
Lines: 14

In article <6rjbp2$118$1@52-a-usw.rb1.blv.nwnexus.net>, tzs@halcyon.com
(Tim Smith) wrote:

> Do you have a cite for that formulation of the test? That doesn't quite
> match any test I recall from Professor Chisum's patent course, but I took
> that a few years ago, and have not kept up with subsequent cases.

35 U.S.C. s. 103 (read in pari materia with 102) and Graham v. Deere?

--
just another view,
Andy Greenberg
(remove "nonspam-" to get my real mailing address)

Subject: Re: Pamela Samuelson panelist at Open Source town meeting 8/21/98 San Jose,
CA
Date: 13 Aug 1998 08:08:46 -0400
From: Craig Burley <burley@tweedledumb.cygnus.com>
Message-ID: <y690ktrsmp.fsf@tweedledumb.cygnus.com>
References: <6qfn8t$sem@bourbon.cs.umd.edu>
<35C74BAC.547245AC@ix.netcom.com>
Newsgroups:
misc.int-property,misc.legal.computing,comp.os.linux.misc,gnu.misc.discuss
Lines: 418

"Karsten M. Self" <kmself@ix.netcom.com> writes:

> You've got some good points, many of them well made.

I must be slipping. :)

> Unlike the designer of a
> physical machine, the software engineer has no standard
> components. She cannot buy off-the-shelf parts or
> sub-assemblies from suppliers. Typically, a programmer
> writes every line of code afresh, no matter how large the
> program is, or how common its tasks.

Nice, but not entirely true, and the real problem will show up early
next century: when the code is no longer imperative, or if it
"looks" that way and is just reverse-engineered into higher-level
abstractions, then the *computer* (think "compiler" if you have to)
chooses the algorithms to apply.

Which means the *computer* violates a patent of some algorithm
applied to some field. But of course, computers can't violate
patents -- only people can, either the people who tell the computer
about the problem to solve, or the ones who taught the computer
to be that smart, or the ones who sold the computer. *None* of
whom have *any* fair opportunity to even realize they've "invented"
the patented "process".

That's what bugs me about, e.g. the "backing store" patent. As a
patent on an interface design it's perhaps conceivably useful (though
not to publish, since the interface is published).

But, as a patent on "gee, we're so smart, we figured out this new
way to save time", it's laughable.

Because, *any* computing engine, given the overall problem -- which
is that a "continuous" function of some degree of complexity takes
some inputs and produces some outputs -- could have reasoned "hmm,
given that the inputs often remain the same over time, and the outputs
can be small in size compared to the complexity of the function,
let's save the outputs even when they're temporarily not needed".

So the problem with the "classic" view of what software "is" will
inevitably lead to John Q. Public "violating" patents by daring
to combine his favorite interface with his favorite spreadsheet,
because his computer then picks a bunch of algorithms that, voila,
violate patents.

To avoid this, we have to avoid designing intelligent software.

Period.

Copyright, on the other hand, seems much further away from the point where this'll become a large-scale problem, because it isn't based on a land-mine model; because independent creation is a defense; and maybe other things.

> The design space a
> programmer must search through to ensure lack of patent infringement is
> vast, and prohibitive to many in the OSS community.

My impression is that it's prohibitive to most of the software industry in general. I've worked shoulder-to-shoulder with "hardware guys" designing new computers and have seen the differences to get at least a feel for how different software really is, and how quickly it moves (or, put another way, how little patentable "up front" design is done, for good or ill ;-) compared to hardware.

> - The patent base is still growing -- floodgates have opened ever wider
> since 1990. Critical mass has not yet occurred.

Well...and the whole industry is burgeoning so quickly, etc. It's really silly to waste any time trying to profit off of a few software patents these days -- one can make so much more money doing Real Work. It's the few who try and become annoying I'm worried about *now*, and it's the future when the industry isn't quite so "paved with gold" and larger portions of it "shrink-wrap" around the software-patent possibilities that I'm also worried about.

That's why I feel, in the long run, it'll all be taken care of.

E.g. just like the guy who trademarked "Linux" got "taken care of". Again, though, he was on vastly shakier legal ground -- think what just ten guys like that holding software patents that are really useful only against OS authors like myself might be able to do, and remember, it's our *personal* fortunes, not those of people using names like "Linux" in print, that are at stake!

In other words, yes, patent, like copyright, is all IP law, but the risks and costs of unknowingly violating patents are *much* higher than with copyright.

> The Open Source Patent Patron Solution

It's nice, but one negative is the same one I see already happening with the justification of software patents -- once implemented, it'll rapidly become a justification for how "important" the software-patent and proprietary-software industry is for OSS to survive.

Which gets back to my point, being, why do we *need* this massive beauracracy on top of this other massive government-sponsored beauracracy? I mean, other than to enrich the few people well-positioned to benefit from it, but not as those who actually forward the technology? It's a value judgement I feel is clearly in favor of chemical-process patents, but almost certainly *not* favorable for software patents. Does *anybody* really believe software patents, even adding the OSPPS you propose, leads to a globally optimal solution?

Put another way: scientists studying flora and fauna seem to have little trouble distinguishing forms that simply leech off of other forms from those that engage in mutually beneficial relationships, but I see plenty of tendency among humans to view any human activity that seems potentially "leech-like" as if it were mutually beneficial, using the line of reasoning that the larger activity is "doing fine" and hence would only be *worse* off otherwise.

So, right now, we're seeing implications posted that, without software patents, products (other than g77) wouldn't exist.

But, strangely, as long as I've worked in the industry, including doing contracts, software patents *never* enter into the equation except as either "feathers in a cap" (not seriously pursued) or for defensive reasons (more seriously pursued, but not reliable for the purpose).

Since I believe patents themselves are *important* to ensure publication, it's just the areas to which they are applied that I question, based on whether their value changes from "net good" to "net bad", or basically leech-like.

So I question whether patent protection should be granted to software, while I don't really when it comes to chemical ("matter-transforming" is my wording) processes. I also question the applicability to genetic codings, but don't know enough about that field to really get into it. (With some awareness of some similarities to neural-net research in software, I have about as much faith in patent suitability for "voila, we finally selected a working gene via repeated random selection" as I do in "voila, after running the neural-net software, we finally got

a net that solves X". Either way, I don't quite see that we need so much protection for the result of giving a big computer a lot of time and memory to work out a problem...but when somebody truly "codes up" DNA from scratch, knowing ahead of time how it'll work, then they're writing software, IMO, and perhaps copyright, though still not patent, protection should apply.)

> Probably true. But the businesses which have survived and/or thrived on
> account of their IP being protected are much less dramatic. Imagine
> what the newspaper would read like if "live notices" were published
> rather than just the obits of those who'd died recently. Andy will
> hammer you one this, I'm just beating his punch.

It'd help if I'd *ever* run into this myself. It's amazing how much this supposedly happens, when I never see it happening, despite having done plenty of proprietary work, involving clearly patentable subject matter.

So, you can imagine how my mind boggles at the thought of software-patenting becoming as widespread as software-*copyrighting* such that every programmer was aware of it. Think of what that'd do to the lead times on patent applications; cross-licensing; and so on.

IMO, it'd destroy the OS (open-source) community in short order, unless widespread legal relief was provided.

The *big* advantage I see to this whole system is that it throws the door wide open to any third-world countries wanting to leapfrog the West (by which I really mean the countries signed onto the various IP protection schemes we use, via treaties). While we're busy scouring every line of code we write for patentability, and every line everyone else writes for infringement, they'll learn to just Write The Doggone Code, release it, and perhaps end up doing to our software industry what the Japanese did to our auto industry in the '70s. (Only I think we'll have much less of leg to stand on, economically speaking, given the fluidity of software development compared to automobile design and production.)

> There is a substantial body of literature suggesting that various forms
> of IP are neutral or even net harmful in their social impacts. One of
> the more interesting such is a 1970 Harvard Law Review paper "The Uneasy
> Case for Copyright". Though it principally covers printed media, it
> also discusses computer software, and patent protection of software,
> finding the prospects of both (neither were then statutory or
> commonplace) disquieting. The author is Steven Breyer. His current job
> is Associate Justice, U.S. Supreme Court. This is a good one to
> brandish when the "lunatic fringe" label is lobbed at you <g>. I'll
> attach a list of economics papers I've turned up online at the end of
> this post, if you're interested in more studies. Interestingly, I
> haven't found any economic research suggesting copyright/patent are
> insufficient or necessary.

I find arguments that software is unique as copyrightable material quite interesting. I just haven't gotten *close* to having enough data to make up my mind. So I challenge "rock-solid" assumptions people make, as I do about how software patents can't really be a drag on the industry. I've seen others do this a few times on this thread vis-a-vis software copyright, which is among many reasons I've skipped doing so myself.

> > > One way to get lawyers interested is to stop bashing them and the system
> > > they address; particularly since the chances of changing the law as
> > > applied is virtually nil.
> >
> > Isn't it amazing how lawyers are always the ones who get to *make*
> > the laws, then afterwards claim "well, you can't change the laws,
> > so don't bother trying -- just make friends with *us* and maybe
> > we won't charge so much for you to conduct business in this
> > environment"?
>
> Craig, he's got a point. He's not necessarily the enemy either. And if
> you can find a solution that guarantees him a paycheck or billable
> hours, you might even get yourself some good counsel.

Right, I know that -- I've already talked to a lawyer about some related issues, because I've got some long-term strategies to deal with this and other problems.

My point is that this whole system becomes self-fulfilling, and I'm simply not going to be impressed when somebody says, in effect, "don't say the emporer has no clothes, just keep paying us for telling him how nice he looks in them".

Remember my entry into this thread: a *very* short post pointing out that the Forbes article on free software could be a useful data point vis-a-vis the theory that software patents can't really slow down an industry that takes tons of resources to produce the typical large program.

That was met with, paraphrased, "but the *imputed* value of that free-software work is very high", at which point I decided to explain that the software patent system, and lawyers in general, don't take payment in imputed value. (I should have said it that succinctly back then. :)

> The simple truth is that though the lawyers write the law, they're the
> lapdogs of the current economic interests. The current economic
> interests are an established technology and proprietary software
> industry which is the current industrial engine of the mightiest economy
> on the planet. It's not the lawyers, it's Microsoft. It's also IBM,
> Sun, Intel, SAS, Oracle, whatever Borland is calling themselves this
> week, etc. When he's not beating us up here, Andy does what he's paid
> to do (when I'm not taking his punches, and occasionally giving back, so
> do I -- programmers are just less-well-dressed lapdogs).

You know, the current economic interests probably want me to design advanced weapons systems that can wipe out nations, rioters, people of certain races, whatever.

Instead, I choose to do what I feel is "most right", a combination of globally optimal economic and market behavior and personal moral convictions.

So, you have to pardon me when I responded unfavorably to someone saying "just accept the system as is, and pay us to help you cope with it -- oh, sorry if you can't afford to while you're making the world a better place, as we have different objectives than you".

It's not so much that they're *wrong* -- I just want it to be clear, more and more, that that's the decision *they're* making, and that it's the ultimate reasoning behind their values.

In particular, I believe it's worthwhile to challenge this notion (which I find entirely unsupportable in my experience) that software patents are in any way crucial for the burgeoning software industry to survive or to flourish. As a programmer, you'd *think* I'd be taken seriously on this; even as a barely-wannabe CPU hardware designer, I have *no problem* seeing the usefulness of patents in that area.

If, in the meantime, I can challenge one or two individuals to ask themselves why they're simply going with the "corporate revenue stream" to guide their public discourse on any issue, fine. Doesn't matter if I change their minds, as long as they might *think* about it. I've changed minds before on some pretty intransigent issues (based on emails I've gotten), which is kinda scary in a way.

> The law hasn't changed (much) since software copyright was instituted in
> 1980. The court's interpretation of what protections are granted by the
> law has changed considerably. The interest in SW patent is in large
> part a response by the SW industry to the increasingly apparent weakness
> of IP protection yielded by copyright as the result of several key court
> decisions (noteably, CA v. Altai, Lotus v. Borland).

That's what I was getting at, with less precision. It doesn't matter *much* whether the law changes, or the interpretations change. Ask me, and you'll hear a much more "Borkian" view about it -- I have this peculiar notion that the courts should provide only a strict interpretation of the existing law, and IMO this should err in the direction of *less* interference in peoples' lives. (Don't think I'm saying this makes judicial work any easier. About all it does is make people think a bit more about whether, and for whom, to vote, since they can influence legislatures. Instead, these days, they just watch Oprah, yell a lot, and hope judges and juries "hear" them somehow...though that does seems to work pretty well. :)

Anyway, if effective copyright for software has been weakening lately, then why can't effective patent law for software do so?

I don't remember voting for *increased* software patentability -- about the closest I came to that was voting for Reagan (twice, but IIRC the first one did the trick on this issue).

So, the PTO and the courts "adjust" their behavior, largely ignoring the role of the legislature(s) to set the rules so we can all participate in this process up front.

Not ideal; but I don't see why software-patent law can't "take a hit" in the future, as the result of all sorts of things, including discussions like this.

(Years ago, I was told "don't bother talking about the viability of free software, such as GNU, because your talking won't help". Well, *something* helped, and I've had people tell me via email how much my posts inspired them to adopt, if not contribute to, things like GNU or Linux.)

> Salient. I think Andy's badgering you a bit much on the distinctions
> between OSS and the SW industry as a whole. It might be the result of
> his having less clouded vision. I rather suspect though that he's a
> behind the curve on really understanding what OSS is about in terms of a
> SW development and an economic model, but he's sharp and might catch on
> soon <g>. When companies like Netscape and IBM step up to bat with OSS
> products, it becomes much harder to draw a distinction between "OSS" on
> the one hand and "the SW industry" on the other -- they are becoming one
> and the same, and will become more so.

Thanks, that was indeed my original point by pointing to the Forbes article.

Part of my problem is that I've had a vision about software (and other types of) engineering for nearly 30 years now. The free-software paradigm was a crucial answer to a very tough question, though not necessarily the only one (however, it *is* almost certainly the least expensive, globally).

So, for nearly 30 years, I've understood how unlike today's software-development paradigm the paradigm of, e.g., the decade starting around 2020 is likely to be. Especially since my "education", properly, is poor, yet my pie-in-the-sky leaps even from my early days are being (somewhat slowly, I admit) fumbled, sometimes leapt, towards by the industry. The things I've been bad at predicting have involved *underestimating* how small, and ubiquitous, machines can be -- e.g. I envisioned something in many ways much more ambitious, yet more hardware-feasible, tolerant, and reliable than today's Internet back around 1975, but that *didn't* include things like streaming video or even interactive global conversations a la IRC, because I didn't envision the necessary hardware or bandwidth (though my vision certainly *accommodated* both, natch).

I can certainly excuse people not seeing my particular vision, and am interested in theirs.

But to see someone say "well, the software industry works *this* way; working this way is so expensive that the software patent system isn't much of a hindrance; therefore software patents are not a hindrance" is something I have to respond to, especially when the evidence is *already* in that the software industry does *not* entirely work "this way", and software patents already *have* been a hindrance (compress, GIF, and so on).

In the end, software patents are valid *only* if they serve the goals of the software industry and the community at large. I see *no* evidence of this substantial enough to counteract the substantial evidence of its hindrances to date, and plenty of reason to expect the hindrances, but not the "helps", to grow.

> Andy: any equivalent of IP covering portions of legal practice? Other
> than trade secrecy, say analog to copyright or patent?

IIRC, there are some interesting analogies that might be found to software *copyright* versus legal-citation copyright (IP), e.g. West Publishing. It's been year since I was even remotely close to being "up" on this.

> <histrionics> don't dismiss me *that* easily!!! I put at least five
> minutes into that idea!!!

Yes, it's a good one, and what might actually happen. I think there might be more "defense by the people" available than would make your scenario necessary, and as I suggested above, I'd rather see *that*, so "the people", rather than "the big corporations with help from software patents" able to take credit for any and all future success of OSS.

> The irony is that the GPL *requires* copyright to work -- the legal
> rights granted copyright owners under the law are the mechanism which
> drives GPL. In the same way, instead of looking for a way to break down
> patent law, we should look for a mechanism which twists it to suit our
> own purposes.

I agree with that reasoning. If I could apply for a software patent as easily, and get it approved as reliably, and be sure it'd defend me against litigation as surely, as is the case for software copyright, we wouldn't be having this discussion!

> I'll take a wild swing at it and say that Andy's remark "Karsten has put
> his finger" indicates at that the idea has passed a basic plausiblity
> test -- he hasn't replied directly to me. I think it's damned crazy
> myself, but it's crazy good, and maybe enough to work.

Well, you outlined the pitfalls, the Things That Need Doing to make it work. What's to argue? Put up or shut up, I say to anyone who wants to do anything about it. :)

> Those economic studies (and a few other misc. papers):

Thanks. I certainly need to do a lot more reading before I start telling anyone I know all about software patents. I hope I haven't come off as if I felt I did -- mostly I was trying to post from personal experience, OSS experience, and awareness of some of the principles of patent law that I've read up on many years ago.
--

"Practice random senselessness and act kind of beautiful."
James Craig Burley, Software Craftsperson burley@gnu.org

Terry Ritter, his current address, and his top page.

Last updated: 1999-02-20

Software, Patents and Algorithms

A *Ciphers By Ritter* Page

"Software patents," starting out with the infamous XOR cursor patent and ending with "what is an algorithm." A discussion of software protection, infringement, and the LZW patent. Here we have patent lawyers and software geeks, and a classic failure to communicate.

Contents

- 1998-08-01 Karsten M. Self: "4,197,590"
- 1998-08-03 Andrew C. Greenberg: "Has anyone considered that the LPF's parade of horribles never happened?"
- 1998-08-3 Tracy Aquilla: "By now, software is not really a new technology anymore, most of the major legal issues have been resolved, and the state of the law is pretty clear."
- 1998-08-03 Bruce Hayden: "...serious software development requires serious financial investment, and software patents provide a mechanism to protect this financial investment."
- 1998-08-03 Karsten M. Self: "The problem from the developer's perspective of XOR is that this is a trivial and possibly non-novel concept now protected by patent -- which addresses a ubiquitous problem in GUI programming."
- 1998-08-04 Victor A. Wagner, Jr.: "Maybe if you'd been around in the 1960's when Raytheon originally filed for the XOR (to generate a 'blinking cursor' you would have done what everyone else I knew in the business did: smack their forhead and make a comment like "F**K, why didn't I think of that". Which makes it non-obvious just in case you missed the point."
- 1998-08-03 Andrew C. Greenberg: "Have you noticed that GUI programming is ubiquitous and there haven't been massive lawsuits? What does this suggest about the so-called XOR patent?"
- 1998-08-03 Andrew C. Greenberg: "LPF in 1991 said patents would bring the software industry to its knees. That didn't happen, and its NOT a matter of opinion."
- 1998-08-04 Rahul Dhesi: "Most commercial software is distributed in binary-only form, so it's quite hard to know what is happening inside that might violate a patent. What we have here is a ticking time-bomb."
- 1998-08-04 Bruce Hayden: "Ever try finding a certain configuration of say six transistors on an IC with six million transistors?"
- 1998-08-04 Andrew C. Greenberg: "...Dhesi's "time bomb" has been 'ticking' for 10 years and not a single glitch. Not bad to disprove the proposition that this would soon bring the software industry to its knees."
- 1998-08-05 Bruce Hayden: "...if you accept that a number of basic concepts have been patented in the software arts due to the PTO's inability to adequately search the patent art..., then many of these basic concept software patents should be coming up on their half life, without having done any noticable damage."
- 1998-08-07 Andrew C. Greenberg: "Manufacturing software is very cheap compared to, say, manufacture of cars -- and it is true that you don't need a factory to do it -- but design and implementation costs have gone well through the roof."
- 1998-08-07 Bruce Hayden: "It is rare to recover attorneys' fees in patent litigation. Thus, to be economically viable, a patent holder should be able to expect to recover his litigation costs through damages in patent litigation. At a minimum, this really only starts happening when infringing sales get into the millions, probably more realistically the tens of millions of dollars."
- 1998-08-08 Andrew C. Greenberg: "Patent infringement is a two-step process: (1) construe (interpret) the claims; and (2) determine whether the claims properly construed 'read on' the accused device."
- 1998-08-10 Andrew C. Greenberg: "No algorithms are protected by patents in the USA, even under State Street Bank. Certain applications of those algorithms most certainly are protected."
- 1998-08-10 Andrew C. Greenberg: "The claims define the invention. You cannot claim an algorithm, although the mentioning of an algorithm or formula as part of a claim does not render the claim invalid."
- 1998-08-12 Olivier Galibert: "Please explain me in which way this is different than 'an algorithm for data compression', except that it avoids said 'algorithm' word."
- 1998-08-12 Andrew C. Greenberg: "The devil is in the details." "Note that the claim is directed to an apparatus, and not an algorithm."
- 1998-08-12 Andrew C. Greenberg: "No doubt methods are patentable, and algorithms, as understood in the technical sense, are very much like methods, understood in the legal sense. But not all algorithms are patentable as methods. This is where confusion seems to reign among lay audiences."
- 1998-08-12 Tracy Aquilla: "Infringement may be found only if the accused device or process meets each and every limitation of at least one claim."
- 1998-08-12 Barry Margolin: "I suspect that someone could easily get a patent on 'a method for reordering data in a computer memory' if the particular sorting algorithm it used were novel and unobvious. And had people been in the habit of patenting computer programs in the 60's, I suspect the inventor of quicksort might have gotten a patent on it."
- 1998-08-13 Andrew C. Greenberg: "Check out the PTO's software patenting guidelines, and reread State Street Bank."
- 1998-08-14 Andrew C. Greenberg: "If you infringe any one claim and the claim is valid, you are infringing."
- 1998-08-12 Tracy Aquilla: "Not only does the claim avoid the word 'algorithm', it is clearly directed to an apparatus, not an algorithm."
- 1998-08-12 Barry Margolin: "All the lawyers seem to be making a distinction that doesn't exist in the real world, just so that they can justify the claim that you can't patent an algorithm."
- 1998-08-13 Andrew C. Greenberg: "...understand that it is not only the manifestation of the algorithm on a machine that limits the patent, but also the particular application or class of application of the algorithm."
- 1998-08-04 Andrew C. Greenberg: "Which of the claims prevent people from reverse engineering MSDOS?"
- 1998-08-04 Rahul Dhesi: "If you read the whole thing, it will become clear to you that the purpose of the STAC patent was to achieve a monopoly on the basic concept of 'when compressing a device, don't change its name.' The claims were simply a means to that end."
- 1998-08-11 Tracy Aquilla: "One might be able to 'get patent protection', but the claims cannot be directed to an algorithm."
- 1998-08-11 Bruce Hayden: "...ALL method claims are algorithms. What is prohibited are 'mathematical' algorithms." "What you have to start with is the realization that a general purpose computer running a particular program is a special purpose computer, which is considered an apparatus, and thus statutory."
- 1998-08-12 Tracy Aquilla: "One cannot infringe an apparatus claim or a method claim by merely using an algorithm that is but a single element of the claim." "The question remains: does this prevent others from using the algorithm itself in other methods or apparatus? I do not believe that it does."
- 1998-08-12 Rahul Dhesi: "The broader claims of patent 4,558,302 claim the LZW algorithm for data comression.... LZW isn't just a proper subset of what is being claimed, LZW *is* what is being claimed...."
- 1998-08-12 Tracy Aquilla: "Well claim 1 is the broadest claim, and it is directed to an apparatus. Perhaps you can explain the basis for your conclusion that the 'broader claims' effectively claim the algorithm itself?"
- 1998-08-12 Rahul Dhesi: "The attorneys for Unisys, which inherited the patent from Univac, have been quoted as saying that all implementations of LZW, whether in hardware or software, are protected by the patent."
- 1998-08-13 Bruce Hayden: "And what is wrong with that?"
- 1998-08-18 Rahul Dhesi: "...paper copies of patented software will not violate the patent, unless such paper copies are functional in such a way that their existence could be considered an infringing 'use' of the patent."
- 1998-08-19 Barry Margolin: "As a customer, should I have to verify that I don't turn my computer into an infringing device? And how am I supposed to do so, since I am not privy to the mechanisms embodied in most of the software I run?"
- 1998-08-21 Andrew C. Greenberg: "Since contributory infringement is an independently actionable tort, I doubt that a direct infringer needs to be named."
- 1998-08-12 Andrew C. Greenberg: "Perhaps LZW is what the authors would have liked to have claimed, but it is far from clear what is the scope of the patent. Many software patents construed in recent Markman hearings have been narrowly limited to particular applications, despite the broad 'algorithm-like' nature of their claim language."
- 1998-08-13 Rahul Dhesi: "Do you believe that there are certain implementations of LZW that are not protected by the patent?"
- 1998-08-13 Andrew C. Greenberg: "The Supreme Court stated many times unequivocally that, 'it is the claim that defines the invention and gives notice to the public of the limits of the patent monopoly....'"

- 1998-08-13 Rahul Dhesi: "If LZW is not itself protected, then it should be possible to find implementations of LZW that do not infringe the patent."
- 1998-08-13 Tracy Aquilla: "To be precise, nothing is actually 'protected' by the patent. Patents do not 'protect' an invention, they merely provide the patentee a legal right that others do not have - the right to file lawsuits for infringement of the patent."
- 1998-08-12 Chris Pastel: "In short, every method patent patents an algorithm, but not a mathematical algorithm."
- 1998-08-12 Andrew C. Greenberg: "...the notion of algorithm, as defined by the Supreme Court *IS* narrower than the notion of an algorithm as Chris is using it here."
- 1998-08-13 Tracy Aquilla: "Apparently the word "algorithm" is derived from the Greek word for 'number'."
- 1998-08-13 Dr. Ram Samudrala: "The word algorithm is derived from the name of the Persian mathematician Mohammed al-Kowarizmi.... When written in Latin, the name became Algorismus...."
- 1998-08-13 Tracy Aquilla: "Your chosen dictionary did not list the Greek source of the spelling of algorithm? Webster's II says: Greek source _arithm_, 'number'." "Apparently, the US law and the PTO use the more narrow definition of algorithm to distinguish 'algorithms' from 'methods', because some methods are patentable, while algorithms are expressly not patentable."
- 1998-08-14 Dr. Ram Samudrala: "I don't care whether algorithms are patentable or not; I ignore the entire body of patent law when it comes to my actions."
- 1998-08-14 Dr. Ram Samudrala: "I just didn't want anyone to think I had an opinion about the patentability of algorithms in this thread."
- 1998-08-14 Andrew C. Greenberg: "My Webster's Third New International... likewise defines algorithm in terms of calculation and arithmetic. Indeed, solely so. Algorithm is defined in terms of the word algorism, which in turn is defined as the art of calculating with the symbols 0-9."
- 1998-08-15 Dr. Ram Samudrala: "You really honestly believe that the word algorithm didn't originate from the name of Mohammed al-Kowarizmi and instead believe it originates from the greek word for number...?"
- 1998-08-15 Tracy Aquilla: "Maybe the 'al' part did?"
- 1998-08-15 Andrew C. Greenberg: "I suggest that those seriously concerned about the Supreme Court's ability to comprehend the oridinary and technical meaning of words consider the footnote in Diamond v. Diehr, which acknowledges broader uses of the term and discusses those meanings in conext of the opinion as a whole."
- 1998-08-13 Bruce Hayden: "Again, it is 'mathematical' algorithms that are not patentable."
- 1998-08-14 Tracy Aquilla: "Do you know of an issued patent wherein the claimed invention is a non-mathematical 'algorithm'...?"
- 1998-08-13 Bruce Hayden: "The term 'mathematical' algorithm in patent law has been signficantly limited to an algorithm where you have a handful of numbers input, and one or a small number of numbers output." "...the big thing that these algorithms (in whatever guise) have is that they relate very strongly to hardware. There must be hardware involved. Software without hardware is still pretty much unpatentable. However, loading it in RAM usually creates a statutory machine. Writing it to a floppy or a CD-ROM usually creates a statutory article of manufacture. Specifying a structural relationship between hardware elements in method claims usually results in a statutory process."
- 1998-08-14 Andrew C. Greenberg:
- 1998-08-14 Chris Pastel: "If you are going to use a word in a legal discussion or argument, then you should abide by the 'legal' meaning of the word and not get wound around the axle by insisting on the 'plain' or 'everyday' meaning."
- 1998-08-16 Bruce Hayden: "...you are correct that we define it almost tautologically - if something is patentable, then it isn't a mathematical algorithm. But we are forced into it because of the Supreme Court cases." "...you are seeing how the law grows over time." "The Supreme Court started the ball rolling in Diamond v. Diehr, where they pointed at the term 'mathematical algorithm' and pointed out that they obviously didn't have such in their case, rather, they had a machine...."
- 1998-08-17 Bruce Hayden: "What you seem to want is a one or two line definition. You aren't going to get one. That is because the term is now a legal term of art. And legal terms of art invariably cannot be defined that cleanly.' "What you have to keep in mind is that the term does have a meaning. It is just that the term stands for the string of court cases."
- 1998-08-12 Andrew C. Greenberg:
- 1998-08-13 Barry Margolin: "So the whole reason for the argument in this thread is a stupid terminology distinction? Can we lay it to rest and agree that it *is* possible to patent a computer algorithm?"
- 1998-08-13 Tracy Aquilla: "Without any convincing evidence? Certainly not."
- 1998-08-13 Andrew C. Greenberg: "No."
- 1998-08-13 Tracy Aquilla: "By using an apparatus that does not meet all of the limitations of the claim."
- 1998-08-13 Tracy Aquilla: "A process that merely manipulates an abstract idea or performs a purely mathematical algorithm is nonstatutory." "For such subject matter to be statutory, the claimed process must be limited to a practical application of the abstract idea or mathematical algorithm in the technological arts."
- 1998-08-14 Rahul Dhesi: "It seems to me that any algorithm that fulfills a need must be considered to be of practical use and hence eligible for patent protection, if it fulfils other requirements such as being new and nonobvious."
- 1998-08-14 Tracy Aquilla: "The MPEP is replete with specific examples...."
- 1998-08-14 Andrew C. Greenberg: "These terms are very well-understood and defined in the references." "The abstract function descibed in this manner is expressly unpatentable."
- 1998-08-14 Olivier Galibert: "Please explain me that, since for the so-called LZW patent nobody in the whole OSS community has been able to come up with a program implementing an algorithm giving the same output for the same set of inputs which wouldn't infringe at least the first claim, pretending that the LZW algorithm itself isn't in fact protected is anything but wordplay?"
- 1998-08-15 Tracy Aquilla: "The precise question is: is there an embodiment of the algorithm that does not fall _within the scope of the claims_?"
- 1998-08-15 Tracy Aquilla: "I believe a standard dictionary of the English language is the more appropriate source for the plain, ordinary meaning...."

```
Subject: XOR: 4,197,590 (was: Re: Patents and OSS)
Date: Sat, 01 Aug 1998 10:12:42 +0000
From: "Karsten M. Self" &LT;kmself@ix.netcom.com>
Message-ID: &LT;35C2EA1A.4711E3ED@ix.netcom.com>
References: &LT;werdna-3007980805340001@tpafx1-31.gate.net>
   &LT;35BC1CCE.C81EA3C8@ix.netcom.com>
Newsgroups: misc.int-property
Lines: 37

Andrew C. Greenberg wrote:
>
> In article &LT;35BC1CCE.C81EA3C8@ix.netcom.com>, kmself@ix.netcom.com wrote:
>


> >  - Are the patents cited good examples?  I believe the XOR
> > window-blanking algorithm may have been successfully challenged.
>
> Cite the patent number.

4,197,590

If you don't mind a anti-IP screed, the League for Programming Freedom
page with some additional information/adgitprop is here:
http://lpf.ai.mit.edu/Patents/

The IBM Patent Search page for this patent:
http://www.patents.ibm.com/details?patent_number=4197590
```

> --
> just another view,
> Andy Greenberg
> (remove "nonspam-" to get my real mailing address)

--
Karsten M. Self (kmself@ix.netcom.com)

What part of "gestalt" don't you understand?
Welchen Teil von "gestalt" verstehen Sie nicht?

web: http://www.netcom.com/~kmself
SAS/Linux: http://www.netcom.com/~kmself/SAS/SAS4Linux.html

3:01am up 49 days, 30 min, 1 user, load average: 1.65, 1.28, 1.18

Subject: Re: XOR: 4,197,590 (was: Re: Patents and OSS)
Date: Mon, 03 Aug 1998 08:26:59 -0400
From: werdna@nonspam-gate.net (Andrew C. Greenberg)
Message-ID: <werdna-03089808265900001@tpafx1-56.gate.net>
References: <35C2EA1A.4711E3ED@ix.netcom.com>
Newsgroups: misc.int-property
Lines: 85

In article <35C2EA1A.4711E3ED@ix.netcom.com>, kmself@ix.netcom.com wrote:

> > > - Are the patents cited good examples? I believe the XOR
> > > window-blanking algorithm may have been successfully challenged.
> >
> > Cite the patent number.
>
> 4,197,590

Thanks, Karsten. Please remind me of the question that was asked: "good examples" of what? I'll take a look.

> If you don't mind a anti-IP screed, the League for Programming Freedom
> page with some additional information/adgitprop is here:
> http://lpf.ai.mit.edu/Patents/

Thanks for reminding me of this. I haven't looked at the site in years. It is interesting to note that the LPF's Conclusion, stated in 1990:

> If nothing is changed, what is now an efficient creative activity will
> become prohibitively expensive. To picture the effects,
> imagine if each square of pavement on the sidewalk had an owner, and
> pedestrians required a license to step on it. Imagine
> the negotiations necessary to walk an entire block under this system. That
> is what writing a program will be like if software
> patents continue. The sparks of creativity and individualism that have
> driven the computer revolution will be snuffed out.

It's been almost ten years since this "call to action" was written. Since that time, the law has been clarified (most recently with State Street Bank), and software invention patents are no longer a LEGAL controversy. The "floodgates" have been open all of that time. Has anyone considered that the LPF's parade of horribles never happened? That the most prominent example of software patent enforcement has been a "right-on" "as-it-should-be" result in the STAC case? That the patent searching system has been getting better and better?

Also, has anone considered the extent to which LPF's assertion that "software is cheap" is true? Does anyone really believe that commercial software can be produced by one or two programmers with about $10K equipment for about $100,000? Is it possible that the fantasy of cheap and good software was just that, a fantasy? [In my experience in the computer game business, those figures would have been a fantasy in the mid-80's].

Once again, my experience with those who challenge the patent system as applied to software inventions come with arguments in three principal areas:

(1) Patents are Bad
(2) Software Patents are Bad; and
(3) Bad Software Patents are Bad.

Noone will ever get any argument from me on Class 3 issues. I have been beating down bad patents, including bad software patents for years (just as I have been helpting to enforce good ones). As to deficiencies in the Patent System as a whole, while I understand some folks have a religious or fundamental political opposition to it, it has worked very well for hundreds of years and as many or more believe it to be a fundamental benefit to society as those who oppose it. I am prepared to defend the Patent System as a whole, and have in other threads, but its simply not an interesting point when conflated with the issues of software patents.

Thus, the fundamentally intersting question to me is this: why are software patents different from other patent issues? If I can be persuaded that software patents are deficient, it would need to be on one of two bases: (A) that the bad software patents are inherent in the system and cannot be rooted out; or

(B) that there is something fundamentally different about software that makes the benefits of the patent system unachievable or exacerbates the harms somehow when patents are applied.

I have found this to be a useful analytical framework for considering arguments made concerning software invention patents. I am not particularly interested in the Class 1 arguments, as they address policies that are well-settled in the United States and are unlikely to be a basis for change in any policies. Nor are Class 3 arguments typically interesting unless they point to failings of the patent system that are inherent because of the nature of software examination, so much as properties of then-existing databases of prior art.

--
just another view,
Andy Greenberg
(remove "nonspam-" to get my real mailing address)

Subject: Re: XOR: 4,197,590 (was: Re: Patents and OSS)
Date: Mon, 3 Aug 98 14:30:44 GMT
From: taquilla@erols.com (Tracy Aquilla)
Message-ID: <taquilla.1252542284I@news.erols.com>
References: <werdna-0308980826590001@tpafx1-56.gate.net>
Newsgroups: misc.int-property
Lines: 50

In Article <werdna-0308980826590001@tpafx1-56.gate.net>,
werdna@nonspam-gate.net (Andrew C. Greenberg) wrote:
>
>Thus, the fundamentally intersting question to me is this: why are
>software patents different from other patent issues? If I can be
>persuaded that software patents are deficient, it would need to be on one
>of two bases: (A) that the bad software patents are inherent in the system
>and cannot be rooted out; or
>(B) that there is something fundamentally different about software that
>makes the benefits of the patent system unachievable or exacerbates the
>harms somehow when patents are applied.

Perhaps it has something to do with the fact that the law was not originally written with software patents in mind? I believe software patents can be analogized with biotech patents in this regard. When the technology was relatively new, the laws were not written to address these technologies directly, because we simply could not anticipate the technology and the legal issues it would bring with its development.

It took time, the resolution of several landmark cases, and some tweaking of the laws over the years to resolve the novel legal issues raised by software patents. In the meantime, examiners and patent professionals alike struggled with the law, and some patents were issued that, in hindsight, probably should not have issued (some were also rejected that should not have been). One can expect a large number of controversies when the law is unsettled like this.

By now, software is not really a new technology anymore, most of the major legal issues have been resolved, and the state of the law is pretty clear. Today, however, biotechnology is in the same situation software was, although the cases coming down from the CAFC are beginning to give form and consistency to the law. Some biotech patents have been issued that probably should not have issued, and some are rejected when they probably should not be. Occasionally, a major case shakes the field and highlights the law. However, while Congress, the PTO, and the courts resolve these issues, there remain some very interesting legal questions regarding biotech patents, and it will take some time to resolve them. This situation seems reminiscent of the situation with software patents.

In summary, some are probably just poor quality work, but many, if not most of the 'bad' patents are probably the result of examiners and practitioners (and the courts) being somewhat uncertain about how the law applies to a particular technology, especially while the technology and the law are co-evolving.
Tracy

Thomas T. Aquilla, Ph.D.
Genetic Engineering and Biotech
Research and Consulting, Ltd.
taquilla@erols.com

Subject: Re: XOR: 4,197,590 (was: Re: Patents and OSS)
Date: Mon, 03 Aug 1998 09:09:20 -0700
From: Bruce Hayden <bhayden@uswest.net>
Message-ID: <35C5E0AF.7709655B@uswest.net>
References: <lpurpleEx4Dyq.3Ln@netcom.com>
<werdna-0308980826590001@tpafx1-56.gate.net>
Newsgroups: misc.int-property
Lines: 42

Lance Purple wrote:

> How much does Linux cost? The GNU tools? The countless other freeware
> tools you can find on the 'Net (POV-Ray, Arachnophilia, GRASS, etc.)
> You may argue about whether these are "commercial" quality, but they
> do exist, and are quite popular in their little niche markets.

>

I think that that is just it. You can indeed write niche products of
less than commercial quality without a large programming investment.
But serious software development requires serious financial investment,
and software patents provide a mechanism to protect this financial
investment.

> These tools were typically created by small groups of unpaid hobbyists,
> who would be instantly bankrupted if a patent suit were filed, even if
> it turned out to be groundless. So far, LPF's fears haven't come true;
> but it is a genuine risk, should Bill Gates et al decide that freeware
> was eating up too much market share and needed to be stamped out.

Of course, the problem here is that MS has not really been
that agressive, at least until the last year or so, with patents.
The result is that that company probably does not have that
strong of a software patent portfolio. So, I would doubt seriously
whether they could shut down for example Linux through
legitimate patent infringment suits.

The flip side of this is that MS is in my mind especially vulnerable
to software patent infringement suits due to their belated interest
in patents and their huge market share.
--

The preceding was not a legal opinion, and is not my employer's.
Original portions Copyright 1998 Bruce E. Hayden,all rights reserved
My work may be copied in whole or part, with proper attribution,
as long as the copying is not for commercial gain.

Bruce E. Hayden bhayden@acm.org
Phoenix, Arizona bhayden@copatlaw.com
***Currently use the following: bhayden@uswest.net

Subject: Re: XOR: 4,197,590 (was: Re: Patents and OSS)
Date: Mon, 03 Aug 1998 18:35:19 +0000
From: "Karsten M. Self" <kmself@ix.netcom.com>
Message-ID: <35C602E7.383E5AD7@ix.netcom.com>
References: <werdna-0308980826590001@tpafx1-56.gate.net>
Newsgroups: misc.int-property
Lines: 227

Andrew C. Greenberg wrote:
>
> In article <35C2EA1A.4711E3ED@ix.netcom.com>, kmself@ix.netcom.com wrote:
>
> > > > - Are the patents cited good examples? I believe the XOR
> > > > window-blanking algorithm may have been successfully challenged.
> > >
> > > Cite the patent number.
> >
> > 4,197,590
>
> Thanks, Karsten. Please remind me of the question that was asked: "good
> examples" of what? I'll take a look.

Jeremy Allison's original statement which I quoted was:

> Don't say there aren't already such patents. The USA patent office will
> issue patents on such things as mathematical formulae (RSA for example)
> and on logical bit operations (the XOR patent that was used to threaten
> the X Windows System).

....indicating, I believe, that XOR is trivial, obvious, and possibly
non-novel.

> > If you don't mind a anti-IP screed, the League for Programming Freedom
> > page with some additional information/adgitprop is here:
> > http://lpf.ai.mit.edu/Patents/
>
> Thanks for reminding me of this. I haven't looked at the site in years.
> It is interesting to note that the LPF's Conclusion, stated in 1990:
>
> > If nothing is changed, what is now an efficient creative activity will
> > become prohibitively expensive. To picture the effects,
> > imagine if each square of pavement on the sidewalk had an owner, and
> > pedestrians required a license to step on it. Imagine
> > the negotiations necessary to walk an entire block under this system. That
> > is what writing a program will be like if software
> > patents continue. The sparks of creativity and individualism that have
> > driven the computer revolution will be snuffed out.

See my post below in this thread WRT cross-licensing -- based on Bruce's
comments. If OSS is as significant as I'd like to believe, it may be
able to get "payment" in the form of cross-licensing protection from
major players who would wield their own patent base to defend OSS. IBM
is one such major player which has taken a significant interest in OSS.

The funny thing about owning a bit of money making property is that you

have to allow others to use it. With sufficient cross licensing, SW patents become almost irrelevent -- I think of analogies to real-estate with private land yet free access on a reciprocating basis. Yes, it's legally possible to play hardball, and a few owners of highly desireable property may be able to command significant access fees, but for the most part, it washes out.

> Also, has anone considered the extent to which LPF's assertion that
> "software is cheap" is true? Does anyone really believe that commercial
> software can be produced by one or two programmers with about $10K
> equipment for about $100,000? Is it possible that the fantasy of cheap
> and good software was just that, a fantasy? [In my experience in the
> computer game business, those figures would have been a fantasy in the
> mid-80's].

Sendmail, Apache, Linux, FreeBSD, the Hurd, the Gimp, BIND, mySQL, KDE, Lyx, emacs, ROOT, mh/xmh, Samba, Mozilla, Tripwire, COPS, Cisco's print-network SW,.... And that's just off the top of my head.

You're wrong on this Andy, but I'll still be your friend. My current axe to grind is that OSS is a large part of SW's future. It may not address all types of SW, but I believe the exceptions will be the minority. The model may also not be the "two programmers with $10k of equipment" scenario -- IMO, much OSS will emerge from corporate and/or academic settings as well. The money and the changes on industry are the interesting parts. Hence my interest in patent.

> Once again, my experience with those who challenge the patent system as
> applied to software inventions come with arguments in three principal
> areas:
>
> (1) Patents are Bad
> (2) Software Patents are Bad; and
> (3) Bad Software Patents are Bad.

The following is a transcript (my tape, my typing) of a question and response at the 7/14/98 Future of Linux Forum. Again, this is a demonstration of the *perception* of patent in the OSS community, not necessarily the actual legal situation. Speakers are a Debian developer, Linus Torvalds, Larry Augustine (runs a HW company, VA Research), and the now familiar Jeremy Allison. Larry is the businessman of the group.

Questioner (Debian developer)

Hi there, I'm a developer for Debian Linux. I maintain a number of packages like the Red Hat folks for things like the Gimp [a graphics tool like Photoshop]. My question is: a lot of users want features for the Gimp like Pantone (r), which is a commercial patented algorithm and set of number for professional printing. I ran into some problems with things like GIF and TIFF algorithms are also patented. We can't distribute them. What does the Open Source world do about software patents, algorithm patents, things like that -- how can OS products like the Gimp compete with Photoshop because they can't make print-ready file submissions [? garbled] -- they're not allowed to be patented?

Responses:

Linus Torvalds:

That's a real can of worms. Patents, especially software patents, are the worst thing that can happen to freely available software. It's just too dangerous to let people do patnets on something that is fundamental research, in many cases, or patent a series of numbers like you mentioned. It does not make sense, and it's sad (?) that it's allowed, and it's a real problem.

The only real solution is to
A). hope that the patent expires which takes a long time or
B). that you create something that is obviously better and patent-free.

And by being patent free you find others who are willing to support your sales (?) and just make sure that maybe Pantone(r) will be there in five years time, but there will be something else too. I'm sorry, there's not much you can do unless you want to spend a lot of money on trying to convince politicians to make certain things illegal.

Larry Augustin (VA Research, Linux International)

I'd just like to ask you to when you run across these issues make them known to people like Linux International (http://www.li.org) and let us know at Linux International if you run into something like that.

We might not be able to help, but on the other hand with maybe ally the vendors like VA, Red Hat, the other people, maybe working together

we can try to help you out.

Jeremy Allison (Samba)

Software patents really are a hidious blot on the software industry, there's no mincing words about it. But I believe that most insane software makers only actaully occur in the U.S. And what this means is that essentially insane software patents end up exporting U.S. jobs. You can make software using the trivial well-known algorithms, they just can't do it in the U.S.

....which may address:

> Thus, the fundamentally intersting question to me is this: why are
> software patents different from other patent issues? If I can be
> persuaded that software patents are deficient, it would need to be on one
> of two bases: (A) that the bad software patents are inherent in the system
> and cannot be rooted out; or
> (B) that there is something fundamentally different about software that
> makes the benefits of the patent system unachievable or exacerbates the
> harms somehow when patents are applied.

My own interpretation:

The problem from the developer's perspective of XOR is that this is a trivial and possibly non-novel concept now protected by patent -- which addresses a ubiquitous problem in GUI programming.

With Pantone, both the maker and users of the GIMP are denied access to several storage/presentation formats which are extremely widespread, industry standard, and in many cases required for consideration of submitted works. I believe the underlying issue is anti-trust or unfair competition -- OSS and its users are placed at an unfair disadvantage by being unable to utilize patented methods -- OSS simply doesn't have the business organization to be able to do so. At least one OSS license, the GPL, makes pretty clear in its preamble that program creators should not apply the GPL to software using patented methods (http://www.fsf.org/). Pantone was also cited as being in part "a series of numbers".

I'm unfamiliar with the specific patent. There is a Pantone website (http://www.pantone.com/) which doesn't reference any patent numbers. IBM patent search suggests:
4878977 http://www.patents.ibm.com/details?&patent_number=4878977
4812899 http://www.patents.ibm.com/details?&patent_number=4812899

....interestingly, these are neither SW nor US patents (inventor is in Germany).

The Pantone page suggest the original patent may be much older, possibly 1963, inventor Lawrence Herbert (http://www.pantone.com/aboutpantone/lh_bio.htm)

In general, many OSS developers have neither the expertise nor financial resources to either ensure that their creations are free of infringing patents or can be defended legally should such a claim be made. Because independent discovery isn't a defense, there is significant concern.

There is also the "offshore" argument made by Allison. SW patents are principally a US phenomenon (AFAIK). Will OSS SW innovation be moved offshore with significant IP barriers in the US? Will the fact that users can be infringers of patent dissuade "deep pockets" users -- major corporations -- from adopting OSS out of liability concerns?

Thanks.

> Andy Greenberg

P.S.: Entsheidungsproblem?
--
Karsten M. Self (kmself@ix.netcom.com)

What part of "gestalt" don't you understand?
Welchen Teil von "gestalt" verstehen Sie nicht?

web: http://www.netcom.com/~kmself
SAS/Linux: http://www.netcom.com/~kmself/SAS/SAS4Linux.html

10:41am up 51 days, 8:10, 3 users, load average: 0.96, 0.50, 0.41

Subject: Re: XOR: 4,197,590 (was: Re: Patents and OSS)
Date: Tue, 04 Aug 1998 00:19:38 GMT
From: "Victor A. Wagner, Jr." <vawjr@home.com>
Message-ID: <35C6538A.1470215B@home.com>
References: <35C602E7.383E5AD7@ix.netcom.com>
Newsgroups: misc.int-property
Lines: 42

-----BEGIN PGP SIGNED MESSAGE-----
Hash: SHA1

Karsten M. Self wrote:
>
> Andrew C. Greenberg wrote:
> >
[deleted]

> Jeremy Allison's original statement which I quoted was:
>
> > Don't say there aren't already such patents. The USA patent office will
> > issue patents on such things as mathematical formulae (RSA for example)
> > and on logical bit operations (the XOR patent that was used to threaten
> > the X Windows System).
>
> ....indicating, I believe, that XOR is trivial, obvious, and possibly
> non-novel.

Maybe if you'd been around in the 1960's when Raytheon originally filed for the XOR (to generate a 'blinking cursor' you would have done what everyone else I knew in the business did: smack their forhead and make a comment like "F**K, why didn't I think of that". Which makes it non-obvious just in case you missed the point.

- --
Victor A. Wagner, Jr.
Candidate for Congress; 47th District, California
Secretary, Orange County Libertarian Party Central Committee
PGP RSA fingerprint = 4D20 EBF6 0101 B069 3817 8DBF C846 E47A
PGP D-H fingerprint = 98BC 65E3 1A19 43EC 3908 65B9 F755 E6F4 63BB 9D93
The five most dangerous words in the English language:
"There oughta be a law"

-----BEGIN PGP SIGNATURE-----
Version: PGP for Personal Privacy 5.5.3

iQA/AwUBNcZTiPdV5vRju52TEQK36QCgyJIHb5C/xZuK3VHEbfSncmzt0OYAn1eH
EjrpFFlfkvsyMjZD5Hzp9/R3
=EzxJ
-----END PGP SIGNATURE-----

Subject: Re: XOR: 4,197,590 (was: Re: Patents and OSS)
Date: Mon, 03 Aug 1998 21:58:38 -0400
From: werdna@nonspam-gate.net (Andrew C. Greenberg)
Message-ID: <werdna-0308982158380001@tpafx1-56.gate.net>
References: <35C602E7.383E5AD7@ix.netcom.com>
Newsgroups: misc.int-property
Lines: 142

In article <35C602E7.383E5AD7@ix.netcom.com>, kmself@ix.netcom.com wrote:

> > Also, has anone considered the extent to which LPF's assertion that
> > "software is cheap" is true? Does anyone really believe that commercial
> > software can be produced by one or two programmers with about $10K
> > equipment for about $100,000? Is it possible that the fantasy of cheap
> > and good software was just that, a fantasy? [In my experience in the
> > computer game business, those figures would have been a fantasy in the
> > mid-80's].
>
> Sendmail, Apache, Linux, FreeBSD, the Hurd, the Gimp, BIND, mySQL, KDE,
> Lyx, emacs, ROOT, mh/xmh, Samba, Mozilla, Tripwire, COPS, Cisco's
> print-network SW,.... And that's just off the top of my head.

Why does everybody presume that I meant by "cheap" or "expensive," the cost to acquire the software once it has been built? I was responding to Allison's assertion that Software is cheap TO MAKE. It isn't, and hasn't been for years.

Everyone of the preceding software packages required many an-years of efforts to create and perfect, and in many cases was made by or derived from substantial government-funded efforts -- often involving more than a few renegate programmers in their garage and in their spare time.

> You're wrong on this Andy, but I'll still be your friend.

Me too, but I don't think I'm wrong. Serious software requires a serious effort, whether the effort is volunteer or otherwise. Allison's "distinction" is no distinction at all, regardless of the price someone may demand for the results of that effort, or who paid to have it created.

> Jeremy Allison (Samba)
>
> Software patents really are a hidious blot on the software industry,
> there's no mincing words about it. But I believe that most insane
> software makers only actaully occur in the U.S. And what this means
> is that essentially insane software patents end up exporting U.S.
> jobs.

Name three jobs that were so exported. The U.S. is one of the most

significant markets for software, and the creation of software abroad doesn't help avoid infringement. A patent is infringed whenever the claimed invention is made, used, sold, offered for sale or imported into the U.S. Sure, if the patent doesn't have a foreign counterpart, the patent can be practiced outside the U.S. for sale into foreign marketplaces; but that presumes there is no foreign counterparts, and that the U.S. market is insignificant to the enterprise or any of its customers.

> You can make software using the trivial well-known
> algorithms,
>   they just can't do it in the U.S.

or sell it in the U.S., or offer it for sale in the U.S., or sell it to someone who will use it in the U.S., or try to import it ino the U.S.

I'm not worried. And serious patents are as likely to have counterparts in most other serious software marketplaces. I suspect that far more jobs are ASSURED in the U.S. because the software patents can be protected here and abroad than are lost because of Allison's claims.

> My own interpretation:
>
> The problem from the developer's perspective of XOR is that this is a
> trivial and possibly non-novel concept now protected by patent -- which
> addresses a ubiquitous problem in GUI programming.

Have you noticed that GUI programming is ubiquitous and there haven't been massive lawsuits? What does this suggest about the so-called XOR patent? Perhaps its scope has been widely exaggerated or its enforceability seriously compromised. I haven't seen us fored back into the land of DOS, have you?

> With Pantone, both the maker and users of the GIMP are denied access to
> several storage/presentation formats which are extremely widespread,
> industry standard, and in many cases required for consideration of
> submitted works. I believe the underlying issue is anti-trust or unfair
> competition -- OSS and its users are placed at an unfair disadvantage by
> being unable to utilize patented methods -- OSS simply doesn't have the
> business organization to be able to do so.

From "The Princess Bride," re: use of "anti-trust or unfair competition."

These words, I do not think they mean what you think they mean.

> Pantone was also cited as being in part "a
> series of numbers".
>
> I'm unfamiliar with the specific patent. There is a Pantone website
> (http://www.pantone.com/) which doesn't reference any patent numbers.
> IBM patent search suggests:
>   4878977 http://www.patents.ibm.com/details?&patent_number=4878977
>   4812899 http://www.patents.ibm.com/details?&patent_number=4812899

My experience is that the great majority of criticisms of software patents involve an equally detailed analysis. I am likewise unfamiliar with the specific patent. Let's look at it before we jump to conclusions, OK?

> The Pantone page suggest the original patent may be much older, possibly
> 1963, inventor Lawrence Herbert
> (http://www.pantone.com/aboutpantone/lh_bio.htm)

That would not be a great threat in 1998.

> In general, many OSS developers have neither the expertise nor financial
> resources to either ensure that their creations are free of infringing
> patents or can be defended legally should such a claim be made. Because
> independent discovery isn't a defense, there is significant concern.

Sure. So you do what all developers do. You get a lawyer and ask. If you are doing it for free, and what you are doing is truly a community service, you get pro bono lawyers to ask. Alterantively, get a sponsor.

Have you noticed that ALL public television and radio programming is copyrighted, and requires licenses to broadcast? This has not prevented PBS from being able to provide its remarkable offerings free to the public.

Don't forget, these things are PROPERTY of others. Tread carefully and do good, if that is what you are doing.
>
> There is also the "offshore" argument made by Allison. SW patents are
> principally a US phenomenon (AFAIK).

Think again.

> Will OSS SW innovation be moved
> offshore with significant IP barriers in the US?

Will the market in the U.S. disappear or benefit from U.S.'s strong IP protection of software inventions? How has that happened in the past. This is a "Patents are Bad" argument, and history indicates otherwise. Weak IP nations have traditionally lagged way behind strong IP nations both in terms of the markets for the IP and the market's ability to create IP.

--
just another view,
Andy Greenberg
(remove "nonspam-" to get my real mailing address)

Subject: Re: XOR: 4,197,590 (was: Re: Patents and OSS)
Date: Mon, 03 Aug 1998 21:34:23 -0400
From: werdna@nonspam-gate.net (Andrew C. Greenberg)
Message-ID: <werdna-0308982134230001@tpafx1-56.gate.net>
References: <6q4s7p$hv5$1@samba.rahul.net>
<werdna-0308980826590001@tpafx1-56.gate.net>
Newsgroups: misc.int-property
Lines: 28

In article <6q4s7p$hv5$1@samba.rahul.net>, c.c.eiftj@12.usenet.us.com
(Rahul Dhesi) wrote:

> In <werdna-0308980826590001@tpafx1-56.gate.net> werdna@nonspam-gate.net
> (Andrew C. Greenberg) writes:
>
> >Has anyone considered
> >that the LPF's parade of horribles never happened? That the most
> >prominent example of software patent enforcement has been a "right-on"
> >"as-it-should-be" result in the STAC case?
>
> This is a matter of opinion. Why should anybody have a monopoly on a
> concept as basic as "when compressing a device don't change its name"?

Rahul likes to say things like "it is a matter of opinion." But it is not
a matter of opinion that the Pollyana LPF claims never occurred: the
software industry was not brought to its knees by rampant patent
infringement actions; free software prospers and by all accounts, the
market has grown enormously and become ever-increasing in importance to
the GNP.

LPF in 1991 said patents would bring the software industry to its knees.
That didn't happen, and its NOT a matter of opinion.

--
just another view,
Andy Greenberg
(remove "nonspam-" to get my real mailing address)

Subject: Re: XOR: 4,197,590 (was: Re: Patents and OSS)
Date: 4 Aug 1998 08:40:13 GMT
From: c.c.eiftj@12.usenet.us.com (Rahul Dhesi)
Message-ID: <6q6hdd$rk1$1@samba.rahul.net>
References: <werdna-0308982134230001@tpafx1-56.gate.net>
Newsgroups: misc.int-property
Lines: 30

In <werdna-0308982134230001@tpafx1-56.gate.net> werdna@nonspam-gate.net
(Andrew C. Greenberg) writes:

>> This is a matter of opinion. Why should anybody have a monopoly on a
>> concept as basic as "when compressing a device don't change its name"?

>Rahul likes to say things like "it is a matter of opinion." But it is not
>a matter of opinion that the Pollyana LPF claims never occurred: the
>software industry was not brought to its knees by rampant patent
>infringement actions; free software prospers and by all accounts, the
>market has grown enormously and become ever-increasing in importance to
>the GNP.

The entire high-tech industry has exploded, probably in a geometric
progression. This observation cannot be used to prove anything about
whether or not software patents are good.

>LPF in 1991 said patents would bring the software industry to its knees.
>That didn't happen, and its NOT a matter of opinion.

Most commercial software is distributed in binary-only form, so it's
quite hard to know what is happening inside that might violate a patent.
What we have here is a ticking time-bomb.

I still want to know why should anybody have a monopoly on a concept as
basic as "when compressing a device don't change its name."
--
Rahul Dhesi <dhesi@spams.r.us.com>
"Frankly, sir, nobody is that much of an expert on anything, at least
not on usenet." -- Andy Greenberg <werdna@gate.net>

Subject: Re: XOR: 4,197,590 (was: Re: Patents and OSS)
Date: Tue, 04 Aug 1998 01:54:50 -0700
From: Bruce Hayden <bhayden@uswest.net>
Message-ID: <35C6CC5A.EA51905A@uswest.net>
References: <6q6hdd$rk1$1@samba.rahul.net>
Newsgroups: misc.int-property
Lines: 34

Rahul Dhesi wrote:

> Most commercial software is distributed in binary-only form, so it's
> quite hard to know what is happening inside that might violate a patent.
> What we have here is a ticking time-bomb.

I seem to have missed the point here. You seem to be arguing
that since it is hard to detect and determine software patent
infringement, that software patents are inherantly bad.
Your original proposition would seem to result in a
contrary result: that software patents are not as valuable
as some others due to these problems.

In any case, software patents are not unique in this respect.
Electronic circuit patents are in my mind probably worse.
Ever try finding a certain configuration of say six transistors
on an IC with six million transistors? It is nearly impossible.

> I still want to know why should anybody have a monopoly on a concept as
> basic as "when compressing a device don't change its name."

Do you have a patent number and claim number that you
are referring to here?
--

The preceding was not a legal opinion, and is not my employer's.
Original portions Copyright 1998 Bruce E. Hayden,all rights reserved
My work may be copied in whole or part, with proper attribution,
as long as the copying is not for commercial gain.

Bruce E. Hayden bhayden@acm.org
Phoenix, Arizona bhayden@copatlaw.com
***Currently use the following: bhayden@uswest.net

Subject: Re: XOR: 4,197,590 (was: Re: Patents and OSS)
Date: Tue, 04 Aug 1998 23:02:09 -0400
From: werdna@nonspam-gate.net (Andrew C. Greenberg)
Message-ID: <werdna-0408982302090001@tpafx1-102.gate.net>
References: <6q6kl5shm1@samba.rahul.net>
<35C6CC5A.EA51905A@uswest.net>
Newsgroups: misc.int-property
Lines: 50

In article <6q6kl5shm1@samba.rahul.net>, c.c.eiftj@12.usenet.us.com
(Rahul Dhesi) wrote:

> In <35C6CC5A.EA51905A@uswest.net> Bruce Hayden <bhayden@uswest.net> writes:
>
> >> Most commercial software is distributed in binary-only form, so it's
> >> quite hard to know what is happening inside that might violate a patent.
> >> What we have here is a ticking time-bomb.
>
> >I seem to have missed the point here. You seem to be arguingthat since
> >it is hard to detect and determine software patent infringement, that
> >software patents are inherantly bad.
>
> My argument was perhaps not very clear. I was responding to a claim
> from Andy. Here is a restatement of what I was saying: "Since most
> commercial software is distributed in binary-only form, many existing
> patents are probably being infringed by commercial software but nobody
> knows it yet. This is a ticking time bomb that will eventually explode,
> as people begin to suspect patent infringement and begin to take steps
> to verify these suspicions."

And Bruce asked you to distinguish this from other industries with similar
circumstances -- indeed from virtually every method of manufacture
patent. Many complex devices are distributed in completed form without a
comprehensive schematic or buildout plan.

Moreover, Dhesi's "time bomb" has been "ticking" for 10 years and not a
single glitch. Not bad to disprove the proposition that this would soon
bring the software industry to its knees. Indeed, the "time bomb" is
merely an attempt to fabricate a non-falsifiable proposition so he could
continue to argue in conspiracy theory fashion the already discredited LPF
party line.

> >> I still want to know why should anybody have a monopoly on a concept as
> >> basic as "when compressing a device don't change its name."
>
> >Do you have a patent number and claim number that you
> >are referring to here?
>
> The infamous STAC patent is numbered 5,414,850. Please see my posting
> <6q5dpb$l84$1@samba.rahul.net> for more information. Claim 1 is
> reproduced below.

It clearly does not create a monopoly no the concept "when compressing a
device don't change its name.

--

just another view,
Andy Greenberg
(remove "nonspam-" to get my real mailing address)

Subject: Re: XOR: 4,197,590 (was: Re: Patents and OSS)
Date: Wed, 05 Aug 1998 00:56:43 -0700
From: Bruce Hayden <bhayden@uswest.net>
Message-ID: <35C8103B.3D544973@uswest.net>
References: <werdna-0408982302090001@tpafx1-102.gate.net>
Newsgroups: misc.int-property
Lines: 39

Andrew C. Greenberg wrote:

> Moreover, Dhesi's "time bomb" has been "ticking" for 10 years and not a
> single glitch. Not bad to disprove the proposition that this would soon
> bring the software industry to its knees. Indeed, the "time bomb" is
> merely an attempt to fabricate a non-falsifiable proposition so he could
> continue to argue in conspiracy theory fashion the already discredited LPF
> party line.

Indeed, if you accept that a number of basic concepts have been patented
in the software arts due to the PTO's inability to adequately search the
patent art (the later at least is a constant refrain from Greg Aharonian),
then many of these basic concept software patents should be coming up
on their half life, without having done any noticable damage. Note too
that this is the point that many patent holders start to drop payment of
maintenance fees on nonperforming patents.

Of course, I don't really accept that premise. Partly this is a result of facing
on more than one occasion basic CS art (such as Knuth) cited by an examiner
as 102 art. Sure, sometimes they miss the basic CS references. But more and
more, as the patent examination corp builds up software expertise, and
CS people continue to be hired as examiners, the basic stuff is missed less
and less.

My statements here are of course limited to my own personal experiences
prosecuting software patents, and the shared personal experiences of
other patent professionals doing the same.
--

The preceding was not a legal opinion, and is not my employer's.
Original portions Copyright 1998 Bruce E. Hayden,all rights reserved
My work may be copied in whole or part, with proper attribution,
as long as the copying is not for commercial gain.

Bruce E. Hayden bhayden@acm.org
Phoenix, Arizona bhayden@copatlaw.com
***Currently use the following: bhayden@uswest.net

Subject: Re: XOR: 4,197,590 (was: Re: Patents and OSS)
Date: Fri, 07 Aug 1998 08:03:14 -0400
From: werdna@nonspam-gate.net (Andrew C. Greenberg)
Message-ID: <werdna-0708980803140001@tstpa1-110.gate.net>
References: <35CA3F79.48F65F4B@shore.net>
<barmar-0608980856260001@barmar.ne.mediaone.net>
Newsgroups: misc.int-property
Lines: 51

In article <35CA3F79.48F65F4B@shore.net>, John W Gintell
<gintell@shore.net> wrote:

> I think that one thing that distinguishes the software industry from
> most others is that the capital investment for inventing software
> is nil and that there are vast numbers of very small enterprises
> (1 person) who are creating software. And with the internet and electronic
> commerce it is economically feasible to distribute this software and get paid
> for it.

This is becoming less and less true. The capital necessary to establish a
competitive and effective start-up venture has grown from NIL (when we
started Andrew Greenberg, Inc. and Sir-Tech from a closet) to millions.
The cost of art alone for most computer games is larger than the annual
revenues of many companies in the garage days. Manufacturing software is
very cheap compared to, say, manufacture of cars -- and it is true that
you don't need a factory to do it -- but design and implementation costs
have gone well through the roof.

> With the rapidly increasing number of software patents it is getting
> more likely that these entrepreneurs are going to start inadvertently
> infringing patents. The large enterprises that hold lots of patents
> will have a pretty good arsenal to go after these people and they might just
> do so. The standard way to fight patent infringement is to have your
own arsenal
> of patents and to file or threaten a countersuit. These small entrepreneurs
> can't spend the money and time to file their own. Further, it is very
> expensive and time consuming to deal with this situation.

This isn't different from other industries. The costs of obtaining patent
protection is small compared to say costs of advertising and costs of

software development. A decent measure for this is that the "rapidly increasing number of software patents" has been going on for more than a decade, yet the parade of horribles simply hasn't happened. Perhaps State Street Bank may encourage patent owners to become more agressive. Perhaps. But we have had Federal Circuit opinions "open the floodgates" before and . . . nothing.

In my relatively limited experience (which is substantialy greater than most), the software patent system has worked well, dealt with real inventions and led to reasonable commercial results. True, businesses have paid royalties for the use of inventions. True, lawsuits have been threatened and asserted. In some cases, they had merit and in others they did not. But I have not seen any shutdown of U.S. industry and in particular, I have not seen any *difference* between those transactions and any other transaction involving patents.

--
just another view,
Andy Greenberg
(remove "nonspam-" to get my real mailing address)

Subject: Re: XOR: 4,197,590 (was: Re: Patents and OSS)
Date: Fri, 07 Aug 1998 07:17:39 -0700
From: Bruce Hayden <bhayden@ieee.org>
Message-ID: <35CB0C83.EA20EAF6@ieee.org>
References: <35CA3F79.48F65F4B@shore.net>
<barmar-0608980856260001@barmar.ne.mediaone.net>
Newsgroups: misc.int-property
Lines: 48

John W Gintell wrote:
>
> Barry Margolin wrote:
>
> I think that one thing that distinguishes the software industry from
> most others is that the capital investment for inventing software
> is nil and that there are vast numbers of very small enterprises
> (1 person) who are creating software. And with the internet and electronic
> commerce it is economically feasible to distribute this software and get paid
> for it.

As noted repeatedly before, this is really false. All you have to do is look at the software development budgets for IBM or Microsoft to see this. Rather, software is extremely labor intensive. Thus, you don't for example need a billion+ dollar fab to play. But being highly labor intensive does not translated into no capital investment. As to your one man operations creating software, yes I am sure that they do that all the time. However, this software is not that patentable for the most part, and is too small to get in the radar of those who do have patent portfolios.

> With the rapidly increasing number of software patents it is getting
> more likely that these entrepreneurs are going to start inadvertently
> infringing patents. The large enterprises that hold lots of patents
> will have a pretty good arsenal to go after these people and they might just
> do so. The standard way to fight patent infringement is to have your own arsenal
> of patents and to file or threaten a countersuit. These small entrepreneurs
> can't spend the money and time to file their own. Further, it is very
> expensive and time consuming to deal with this situation.

The reality is that for the most part, patent holders are not going to go after the one man operations. This is purely financial. It is rare to recover attorneys' fees in patent litigation. Thus, to be economically viable, a patent holder should be able to expect to recover his litigation costs through damages in patent litigation. At a minimum, this really only starts happening when infringing sales get into the millions, probably more realistically the tens of millions of dollars.
--

The preceding was not a legal opinion, and is not my employer's.
Original portions Copyright 1998 Bruce E. Hayden,all rights reserved
My work may be copied in whole or part, with proper attribution,
as long as the copying is not for commercial gain.

Bruce E. Hayden bhayden@acm.org
Phoenix, Arizona bhayden@copatlaw.com
***Currently use the following: bhayden@uswest.net

Subject: Re: XOR: 4,197,590 (was: Re: Patents and OSS)
Date: Sat, 08 Aug 1998 08:32:46 -0400
From: werdna@nonspam-gate.net (Andrew C. Greenberg)
Message-ID: <werdna-0808980832460001@tstpa1-18.gate.net>
References: <barmar-0708982157530001@barmar.ne.mediaone.net>
<6qe0famif1@samba.rahul.net>
Newsgroups: misc.int-property
Lines: 78

In article <barmar-0708982157530001@barmar.ne.mediaone.net>,
barmar@bbnplanet.com (Barry Margolin) wrote:

> In article <6qe0famif1@samba.rahul.net>, c.c.eiftj@12.usenet.us.com

> (Rahul Dhesi) wrote:
>
> >If you have an algorithm for doing something that a court finds to be
> >'equivalent' to a patented method, even if they are not identical, you
> >will likely be found to infringe.
>
> I find that very difficult to believe.

OK, here's how the system works:

Patent infringement is a two-step process: (1) construe (interpret) the claims; and (2) determine whether the claims properly construed "read on" the accused device. Step 1 (called claim construction) is a very elaborate process, somewhat more complex for software patents because (1) a determination needs to be made whether functional langauge in the claim is subject to 112(6) and if so, what is the corresponding structure, and if not, whether what remains is valid under 112(1)-(2).

Thus, the claim is the definition that Rahul has been so worried about "finding."

Step 2 entailes checking to see if something reads on. This simply means to determine whether EACH AND EVERY element and limitation of the claim, as properly construed, is found in the accused device/method. The claim. The presence of additional elements will not avoid infringement, and the absence of a single element requires finding no literal infringement.

However, if the claim is NOT found to read on the accused device, necessarily because one or more elements or limitations are mising, the accused device will be found infringing under something called the Doctrine of equivalents if for EACH AND EVERY missing limitation, the accused device has structure that is equivalent. This is NOT THE SAME AS SAYING THAT THE ALGORITHMS ARE EQUIVALENT -- TESTING FOR OVERALL EQUIVALENCY, SAY, AS TO FUNCTION IS ERROR AS A MATTER OF LAW. Equivalents for this test are determined LIMITATION BY LIMITATION.

A general rule of thumb is that a limitation is equivalent to particular structure in an accused device if the structure performs substantial the same function in substantially the same way to obtain substantially the same result. Other tssts are sometimes used.

There are strict legal limitations on how the DOE is applied: (1) DOE cannot be used to expand the scope of claims so as to read on prior art; (2) DOE cannot be used to expand the scope of claims to reach subject matter disclaimed by amendment or argument during prosecution of the patents; (3) DOE cannot be used to effectively "read out" a limitation of the patent -- EVERY LIMITATION MUST STILL BE PRESENT, LITERALLY OR EQUIVALENTLY, AND THE INVENTIONS CANNOT BE COMPARED "AS A WHOLE"; (4) DOE cannot be used to read a limitation to mean something the claim expressly disclaims ("above the slot" cannot mean below the slot); and (5) DOE cannot be used to read a claim on a variation different from claim language where the limiting aspects of the claim language would have been foreseeable to the applicant during prosecution. CAVEAT: These are super-casual accounts of very complicated doctrine, and these are not all of the legal limitations.

The underlying purpose of DOE is to avoid "a fraud upon the patent," resulting from mere linguistic wordplay when the substance of a limitation is taken or used. A typical equivalent is the use of a substitute well-known in the art, for example, using a transistor in lieu of a relay in a circuit to avoid a claim directed to a relay, when none of the other limitations would preclude its application. [For example, this would prevail under DOE prior to the invention of the transistor, but might not afterwards, since the substitution is a foreseeable variation.]

Clear as mud, right? Sorry, its a LOT of law necessarily summarized tersely here. The bottom line is that Judge Dhesi's use of the language here would get him reversed, perhaps with a giggle, by any court of competent jurisdiction.

--
just another view,
Andy Greenberg
(remove "nonspam-" to get my real mailing address)

Subject: Re: XOR: 4,197,590 (was: Re: Patents and OSS)
Date: Mon, 10 Aug 1998 07:59:53 -0400
From: werdna@nonspam-gate.net (Andrew C. Greenberg)
Message-ID: <werdna-1008980759530001@tstpa1-63.gate.net>
References: <6ql2n9amf1@samba.rahul.net>
<werdna-0908981429360001@tstpa1-61.gate.net>
Newsgroups: misc.int-property
Lines: 99

In article <6ql2n9amf1@samba.rahul.net>, c.c.eiftj@12.usenet.us.com
(Rahul Dhesi) wrote:

> In <werdna-0908981429360001@tstpa1-61.gate.net> werdna@nonspam-gate.net
> (Andrew C. Greenberg) writes:
>
> >Rahul doesn't get it. Algorithms aren't protected, inventions are.
> >Inventions are defined by the claims, not by a definition of an algorithm
> >(however expressed) or a program (however expressed).
>

> 'Algorithm' and 'invention' are not mutually exclusive terms. Every new
> algorithm is an invention. Almost every 'method' or 'process' patent
> includes an algorithm somewhere.

Rahul's wordplays still don't change the fact that his postings have nothing to do with an infringement analysis, and hence nothing to do with the patent system.

> Many algorithms are protected by patents in the USA. Some of the
> better-known ones often discussed on Usenet are RSA encryption, IDEA
> encryption, and LZW data compression.

No algorithms are protected by patents in the USA, even under State Street Bank. Certain applications of those algorithms most certainly are protected. Of course, these wordplays still don't change the fact that Rahul's original posting has nothing to do with an infringement analysis, and hence nothing to do with the patent system.

> Andrew is incorrect in stating that 'Inventions are defined by the
> claims'. (I'm assuming that he was referring to 'patented inventions'.)
> The inventor filing for a patent is his own lexicographer, and the
> meanings of the terms he uses in the claims must be derived by a reading
> of the entire patent.

It is true that ambiguities in patent claims are construed in view of the specification. Indeed, 35 USCA 112(6) requires that certain claims be construed in view of the particular structure or methods set forth in the specification. These aids in construing the claims do not change the fact that inventions are determined by the claims. Other information is likewise used in claim construction, such as the prosecution history and in some cases extrinsic evidence, although extrinsic evidence is not resorted to except to resolve ambiguities.

None of this changes the fact that once the claims are construed, nothing else is used in an infringement analysis: ever. Accordingly, these wordplays still don't change the fact that Rahul's original posting has nothing to do with an infringement analysis, and hence nothing to do with the patent system.

> It would be nice if the claims alone served to distinguish what is being
> claimed as new from what is acknowledged to be prior art.

It's nice.

> But apart
> from the question of terminlogy (above), there is also this: If
> an element in a claim is expressed solely as a means for performing a
> function, then the rest of the patent must be read to determine what
> specific structure of the means is in the scope of the patent. You can
> get the official wording for this from the last paragraph in:
>
> http://www.law.cornell.edu/uscode/35/112.shtml

Which section 35 U.S.C. s. 112 has been cited above and several times by me and others in this thread. Indeed, in previous debates, Rahul willfully ignored this fact when making arguments that particular patents were broader than they seemed. Of course, Rahul reads these statutes with abandon, fantasizing what they would mean. Had he studied the case law, he would realize that Section 112, paragraph 6 does not redefine the meaning of the claim, but narrows its scope to read only upon those particular means for performing the function and their equivalents. It can only serve to narrow a claim, and never to broaden it. Likewise with any construction of a patent in view of the specification.

Rahul still doesn't get it. Nothing but the claim as construed in part 1 of the infringement analysis is used in part 2 of the analysis (the actual comparison of the products with the patent). It is NEVER EVER EVER permissible to compare accused products with commercial products or products with specific embodiments of an invention. NEVER. NEVER. NEVER. Period. Only the claim as properly construed is compared against the accused device. Always. In this sense, the claims define the invention, as I described above.

Only claims are compared to programs to determine infringement. Only claims are compared to prior art to determine validity. End of story.

Nothing that Rahul posted here contradicts the preceding posting discussing infringement analyses. To the extent he didn't entirely butcher and misunderstand the law, his discussion of 35 USC 112 and the lexicographer rule apply only to step 1.

Accordingly, none of Rahul's wordplay here addresses the fact that nothing in his original postings relate at all to an infringement analysis, and hence nothing to do with the patent system.

--
just another view,
Andy Greenberg
(remove "nonspam-" to get my real mailing address)

Subject: Re: XOR: 4,197,590 (was: Re: Patents and OSS)
Date: Mon, 10 Aug 1998 22:10:20 -0400
From: werdna@nonspam-gate.net (Andrew C. Greenberg)

Message-ID: <werdna-1008982210200001@tstpa1-41.gate.net>
References: <6qn7tmfrm1@samba.rahul.net>
<werdna-1008980759530001@tstpa1-63.gate.net>
Newsgroups: misc.int-property
Lines: 100

In article <6qn7tmfrm1@samba.rahul.net>, c.c.eiftj@12.usenet.us.com
(Rahul Dhesi) wrote:

> >No algorithms are protected by patents in the USA, even under State Street
> >Bank. Certain applications of those algorithms most certainly are
> >protected....
>
> For example, if an 'algorithm for data compression' is protected by a
> patent, then only its application to data compression is protected.

I am unaware of any claim ever granted along the following lines:

I claim:

1. An algorithm for data compression.

If Rahul cites one, I'll change my tenor. But until then, it is clear he
does not get it (which means he simply doesn't care shat is the truth --
since noone who has read this thread could responsibly pretend they do not
get it). The claims define the invention. You cannot claim an algorithm,
although the mentioning of an algorithm or formula as part of a claim does
not render the claim invalid.

> >It is true that ambiguities in patent claims are construed in view of the
> >specification. Indeed, 35 USCA 112(6) requires that certain claims be
> >construed in view of the particular structure or methods set forth in the
> >specification. These aids in construing the claims do not change the fact
> >that inventions are determined by the claims. Other information is
> >likewise used in claim construction, such as the prosecution history and
> >in some cases extrinsic evidence, although extrinsic evidence is not
> >resorted to except to resolve ambiguities.
>
> So the claims must be understood in the context of the rest of the
> patent. If the meaning of the claims is in dispute, one would of course
> need to consult the rest of the patent, and keep doing so until it has
> been agreed what meaning shall be assigned to the claims. Recent legal
> precedence is that the judge is the final arbiter of what the claims
> mean. But it is to be hoped that the judge will first listen to
> arguments from both sides, during which arguments the entire patent, and
> not just the claims alone, will be discussed. Andy is trying to
> downplay the fact that a substantial portion of a patent dispute could
> well be about how the claims are to be interepreted.

No, not at all. Claim interpretation is the first part of the two-part
analysis I mentioned earlier. Indeed, the two-part test was most recently
recited in the leading case in this arena: Markman v. Westview
Instruments. All this, of course, is wordplay intended to divert
attention from the fact that Rahul's original posting was based on a
confused and irrelevant view of what DOES, in fact, determine patent
infringement.

> (Once the meaning of the claims has been determined then a jury might be
> asked decide whether an existing product infringes the claims.)

No, they will be asked whether the claims, as the judge has explained them
to the jury read upon the accused device. In other words, they make the
decision PRECISELY as I described in my earlier posting.

> Of course, none of this has much to do with the fact that the Patent
> Office must, as part of its job, compare multiple algorithms to
> determine whether one is 'equivalent' in some sense to another. Which
> is why how fast such comparisons can be done is an interesting question.

Rahul lies. The patent office, as part of its jobs, must compare the
CLAIMS of each patent application against the prior art. Noone has
forgotten that Rahul was told this at the beginning of the thread.

> The Patent Office does not look for infringement, at least not during
> the patent filing process; it looks for prior art, which is a slightly
> different thing.

No, they look for validity, which is a slightly different, but not
substantially different process. The CLAIMS still define the invention,
claim construction proceeds as before, although the PTO is supposed to
take the broadest reasonable construction when deciding patentability.
The issue then is whether the CLAIMS, which define the invention read on
(same as with infringement), the prior art (instead of the accused
device).

> >None of this changes the fact that once the claims are construed, nothing
> >else is used in an infringement analysis: ever. Accordingly, these
> >wordplays still don't change the fact that Rahul's original posting has
> >nothing to do with an infringement analysis, and hence nothing to do with
> >the patent system.
>
> Does anybody know which of my 500 original postings Andy is referring
> to? Was it the one where I said the it took exponential time to
> compare two algorithms and determine if they were 'equivalent' in the
> sense in which the Patent Office is interested?

OK, I'll use this one instead. First, exponential time is insufficient to solve an undecidable problem (the equivalence of two algorithms). Second, a patent examiner doesn't compare algorithms in analyzing a patent application. They read the patent claims as applied for against the prior art. Duh again.

--
just another view,
Andy Greenberg
(remove "nonspam-" to get my real mailing address)

Subject: Re: XOR: 4,197,590 (was: Re: Patents and OSS)
Date: 12 Aug 1998 00:39:14 GMT
From: galibert@pobox.com (Olivier Galibert)
Message-ID: <slrn6t1p1i.aei.galibert@renaissance.loria.fr>
References: <werdna-1008982210200001@tstpa1-41.gate.net>
Newsgroups: misc.int-property
Lines: 30

In article <werdna-1008982210200001@tstpa1-41.gate.net>, Andrew C. Greenberg wrote:
>In article <6qn7tmfrm1@samba.rahul.net>, c.c.eiftj@12.usenet.us.com
>(Rahul Dhesi) wrote:
>
>> >No algorithms are protected by patents in the USA, even under State Street
>> >Bank. Certain applications of those algorithms most certainly are
>> >protected....
>>
>> For example, if an 'algorithm for data compression' is protected by a
>> patent, then only its application to data compression is protected.
>
>I am unaware of any claim ever granted along the following lines:
>
> I claim:
>
> 1. An algorithm for data compression.

I claim:
1. In a data compression and data decompression system, compression apparatus for compressing a stream of data character signals into a compressed stream of code signals, said compression apparatus comprising [...]

Please explain me in which way this is different than "an algorithm for data compression", except that it avoids said "algorithm" word.

This is the first claim of patent 4558302, the LZW one.

OG.

Subject: Re: XOR: 4,197,590 (was: Re: Patents and OSS)
Date: Wed, 12 Aug 1998 07:31:00 -0400
From: werdna@nonspam-gate.net (Andrew C. Greenberg)
Message-ID: <werdna-1208980731000001@tstpa2-11.gate.net>
References: <slrn6t1p1i.aei.galibert@renaissance.loria.fr>
Newsgroups: misc.int-property
Lines: 39

In article <slrn6t1p1i.aei.galibert@renaissance.loria.fr>, galibert@pobox.com (Olivier Galibert) wrote:

> In article <werdna-1008982210200001@tstpa1-41.gate.net>, Andrew C. Greenberg wrote:
> >In article <6qn7tmfrm1@samba.rahul.net>, c.c.eiftj@12.usenet.us.com
> >(Rahul Dhesi) wrote:
> >
> >> >No algorithms are protected by patents in the USA, even under State Street
> >> >Bank. Certain applications of those algorithms most certainly are
> >> >protected....
> >>
> >> For example, if an 'algorithm for data compression' is protected by a
> >> patent, then only its application to data compression is protected.
> >
> >I am unaware of any claim ever granted along the following lines:
> >
> > I claim:
> >
> > 1. An algorithm for data compression.
>
> I claim:
> 1. In a data compression and data decompression system, compression apparatus for compressing a stream of data character signals into a compressed stream of code signals, said compression apparatus comprising [...]
>
> Please explain me in which way this is different than "an algorithm for data compression", except that it avoids said "algorithm" word.
>
> This is the first claim of patent 4558302, the LZW one.

The devil is in the details. The elipsis in particular. Note that the claim is directed to an apparatus, and not an algorithm.

--
just another view,
Andy Greenberg
(remove "nonspam-" to get my real mailing address)

Subject: Re: XOR: 4,197,590 (was: Re: Patents and OSS)
Date: Wed, 12 Aug 1998 11:45:58 -0400
From: werdna@nonspam-gate.net (Andrew C. Greenberg)
Message-ID: <werdna-1208981145580001@tstpa2-11.gate.net>
References: <slrn6t3669.dsf.galibert@renaissance.loria.fr>
<werdna-1208980731000001@tstpa2-11.gate.net>
Newsgroups: misc.int-property
Lines: 59

In article <slrn6t3669.dsf.galibert@renaissance.loria.fr>,
galibert@pobox.com (Olivier Galibert) wrote:

> In article <werdna-1208980731000001@tstpa2-11.gate.net>, Andrew C.
Greenberg wrote:
> >In article <slrn6t1p1i.aei.galibert@renaissance.loria.fr>,
> >galibert@pobox.com (Olivier Galibert) wrote:
> >> This is the first claim of patent 4558302, the LZW one.
> >
> >The devil is in the details. The elipsis in particular. Note that the
> >claim is directed to an apparatus, and not an algorithm.
>
> Ok. Then imagine that I write a program incorporating the underlying
> algorithm in order to, say, create image files. Do I infringe the
> patent? If yes, in which way it is different than actually having the
> algorithm itself protected?

Just as it is important to understand the difference between a program and an algorithm, so too it is important to understand the difference between a program and a patent claim. A discussion of a computer program may support a patent claim and with means plus function claims additionally limit the scope of a patent to taht particular program structure, but a program is not a claim.

The claim defines the invention, no more and no less. Each and every limitation in the claim must be satisfied in order to have an infringement. Accordingly, this discussion in the abstract will inevitably lead to the dissatisfying truth that algorithms, per se are unpatentable, yet broad coverage of a patent claim might reach the scope of algorithm-like inventions. The difficulty lies precisely in discussions in terms of generalities.

No doubt methods are patentable, and algorithms, as understood in the technical sense, are very much like methods, understood in the legal sense. But not all algorithms are patentable as methods. This is where confusion seems to reign among lay audiences.

I suggest reading state street bank, which I provided pointers to earlier on, and which should give a good sense as to what is and is not protectable. Patent protections for inventions that use algorithms is not even legally controversial if the invention is new, useful and unobviuos. Patent protection of claims that might reach any application of a law of nature or mathematical algorithm are far more controversial under Section 101, even in view of state street bank.

No, you cannot get a patent on the law of gravity in the abstract. Yes, you can get a patent on a pendulum. No, you cannot get a patent on quicksort in the abstract. Yes, you can get an apparatus for managing the floor of a stock exchange which uses general purpose computers that employ quicksort to sort stock prices. (provided that such things were novel, useful and unobvious when applied for; unlikely in the given scenarios).

The point is that novelty and infringement determinations are not determined by "comparing algorithms."

--
just another view,
Andy Greenberg
(remove "nonspam-" to get my real mailing address)

Subject: Re: XOR: 4,197,590 (was: Re: Patents and OSS)
Date: Wed, 12 Aug 98 16:22:56 GMT
From: taquilla@erols.com (Tracy Aquilla)
Message-ID: <taquilla.1253326616F@news.erols.com>
References: <slrn6t3h1n.ekk.galibert@renaissance.loria.fr>
<werdna-1208981145580001@tstpa2-11.gate.net>
Newsgroups: misc.int-property
Lines: 19

In Article <slrn6t3h1n.ekk.galibert@renaissance.loria.fr>,
galibert@pobox.com (Olivier Galibert) wrote:
>In article <werdna-1208981145580001@tstpa2-11.gate.net>, Andrew C. Greenberg
wrote:
>>The claim defines the invention, no more and no less. Each and every
>>limitation in the claim must be satisfied in order to have an

>>infringement. [...]
>
>IANAL, so please enlighten me. To infrige a patent, do you have to
>infringe one claim, "enough" claims, or all of them?

Infringement may be found only if the accused device or process meets each
and every limitation of at least one claim.
Tracy

Thomas T. Aquilla, Ph.D.
Genetic Engineering and Biotech
Research and Consulting, Ltd.
taquilla@erols.com

Subject: Re: XOR: 4,197,590 (was: Re: Patents and OSS)
Date: Wed, 12 Aug 1998 23:45:12 -0400
From: barmar@bbnplanet.com (Barry Margolin)
Message-ID: <barmar-1208982345130001@barmar.ne.mediaone.net>
References: <werdna-1208981145580001@tstpa2-11.gate.net>
Newsgroups: misc.int-property
Lines: 20

In article <werdna-1208981145580001@tstpa2-11.gate.net>,
werdna@nonspam-gate.net (Andrew C. Greenberg) wrote:

>No, you cannot get a patent on the law of gravity in the abstract. Yes,
>you can get a patent on a pendulum. No, you cannot get a patent on
>quicksort in the abstract. Yes, you can get an apparatus for managing the
>floor of a stock exchange which uses general purpose computers that employ
>quicksort to sort stock prices. (provided that such things were novel,
>useful and unobvious when applied for; unlikely in the given scenarios).

I suspect that someone could easily get a patent on "a method for
reordering data in a computer memory" if the particular sorting algorithm
it used were novel and unobvious. And had people been in the habit of
patenting computer programs in the 60's, I suspect the inventor of
quicksort might have gotten a patent on it.

--
Barry Margolin, barmar@bbnplanet.com
GTE Internetworking, Cambridge, MA
Support the anti-spam movement; see <http://www.cauce.org/>

Subject: Re: XOR: 4,197,590 (was: Re: Patents and OSS)
Date: Thu, 13 Aug 1998 08:01:57 -0400
From: werdna@nonspam-gate.net (Andrew C. Greenberg)
Message-ID: <werdna-1308980801570001@tstpa2-80.gate.net>
References: <barmar-1208982345130001@barmar.ne.mediaone.net>
Newsgroups: misc.int-property
Lines: 22

In article <barmar-1208982345130001@barmar.ne.mediaone.net>,
barmar@bbnplanet.com (Barry Margolin) wrote:

> I suspect that someone could easily get a patent on "a method for
> reordering data in a computer memory" if the particular sorting algorithm
> it used were novel and unobvious. And had people been in the habit of
> patenting computer programs in the 60's, I suspect the inventor of
> quicksort might have gotten a patent on it.

Suspicions are a poor substitute for arguments. Check out the PTO's
software patenting guidelines, and reread State Street Bank.

An important observation is that while software inventions are clearly
patentable, the scope of allowable inventions is still unclear, and may
well be limited by 101. Although it is true that reciting an apparatus
running a program may be patentable in certain cases, it is unclear how
broad or narrow is the scope of that claim.

--
just another view,
Andy Greenberg
(remove "nonspam-" to get my real mailing address)

Subject: Re: XOR: 4,197,590 (was: Re: Patents and OSS)
Date: Fri, 14 Aug 1998 08:42:43 -0400
From: werdna@nonspam-gate.net (Andrew C. Greenberg)
Message-ID: <werdna-1408980842430001@tstpa1-20.gate.net>
References: <slrn6t3h1n.ekk.galibert@renaissance.loria.fr>
<werdna-1208981145580001@tstpa2-11.gate.net>
Newsgroups: misc.int-property
Lines: 31

In article <slrn6t3h1n.ekk.galibert@renaissance.loria.fr>,
galibert@pobox.com (Olivier Galibert) wrote:

> In article <werdna-1208981145580001@tstpa2-11.gate.net>, Andrew C.
Greenberg wrote:
> >The claim defines the invention, no more and no less. Each and every
> >limitation in the claim must be satisfied in order to have an

> >infringement. [...]
>
> IANAL, so please enlighten me. To infrige a patent, do you have to
> infringe one claim, "enough" claims, or all of them?

If you infringe any one claim and the claim is valid, you are infringing. You infringe a claim if and only if the accused apparatus, manufacture or method embodies EACH AND EVERY ONE of the elements and limitatios of that claim. Generally, the presence of additioal elements will not affect infringement. So, if a patent stated:

I claim:

1. an apparatus comprising an A, B and C.
2. an apparatus comprising an A, Z and D.

No combintation of an A, B and Z will infringe unless the C or D are present. On the other hand, an A+B+C+anything else.will infringe (claim 1, and hence) the patent, whether or not a Z or D are present.

--
just another view,
Andy Greenberg
(remove "nonspam-" to get my real mailing address)

Subject: Re: XOR: 4,197,590 (was: Re: Patents and OSS)
Date: Wed, 12 Aug 98 15:41:14 GMT
From: taquilla@erols.com (Tracy Aquilla)
Message-ID: <taquilla.1253324114C@news.erols.com>
References: <slrn6t1p1i.aei.galibert@renaissance.loria.fr>
Newsgroups: misc.int-property
Lines: 39

In Article <slrn6t1p1i.aei.galibert@renaissance.loria.fr>,
galibert@pobox.com (Olivier Galibert) wrote:
>In article <werdna-1008982210200001@tstpa1-41.gate.net>, Andrew C. Greenberg
wrote:
>>In article <6qn7tmfrm1@samba.rahul.net>, c.c.eiftj@12.usenet.us.com
>>(Rahul Dhesi) wrote:
>>
>>> >No algorithms are protected by patents in the USA, even under State Street
>>> >Bank. Certain applications of those algorithms most certainly are
>>> >protected....
>>>
>>> For example, if an 'algorithm for data compression' is protected by a
>>> patent, then only its application to data compression is protected.
>>
>>I am unaware of any claim ever granted along the following lines:
>>
>> I claim:
>>
>> 1. An algorithm for data compression.
>
>I claim:
> 1. In a data compression and data decompression system,
>compression apparatus for compressing a stream of data character
>signals into a compressed stream of code signals, said compression
>apparatus comprising [...]
>
>Please explain me in which way this is different than "an algorithm
>for data compression", except that it avoids said "algorithm" word.

Not only does the claim avoid the word "algorithm", it is clearly directed to an apparatus, not an algorithm. Even if the algorithm is an element of the claim, the mere use of the algorithm itself cannot infringe the claim, rather the accused device must meet each and every imitation of the claim.
Tracy

Thomas T. Aquilla, Ph.D.
Genetic Engineering and Biotech
Research and Consulting, Ltd.
taquilla@erols.com

Subject: Re: XOR: 4,197,590 (was: Re: Patents and OSS)
Date: Wed, 12 Aug 1998 23:49:51 -0400
From: barmar@bbnplanet.com (Barry Margolin)
Message-ID: <barmar-1208982349510001@barmar.ne.mediaone.net>
References: <taquilla.1253324114C@news.erols.com>
Newsgroups: misc.int-property
Lines: 20

In article <taquilla.1253324114C@news.erols.com>, taquilla@erols.com
(Tracy Aquilla) wrote:

>Not only does the claim avoid the word "algorithm", it is clearly directed
>to an apparatus, not an algorithm.

But if the only practical way to use the algorithm is by embodying it in an apparatus (a computer), what's the difference? All the lawyers seem to be making a distinction that doesn't exist in the real world, just so that they can justify the claim that you can't patent an algorithm. If you can patent a computer running a program that implements an algorithm, haven't

you effectively patented the algorithm? As I said in another post, the only way it seems that someone could use the algorithm without infringing the patent would be on pencil and paper (or in one's head), which is not likely to be practical for any algorithm worth patenting.

--
Barry Margolin, barmar@bbnplanet.com
GTE Internetworking, Cambridge, MA
Support the anti-spam movement; see <http://www.cauce.org/>

Subject: Re: XOR: 4,197,590 (was: Re: Patents and OSS)
Date: Thu, 13 Aug 1998 07:58:30 -0400
From: werdna@nonspam-gate.net (Andrew C. Greenberg)
Message-ID: <werdna-1308980758300001@tstpa2-80.gate.net>
References: <barmar-1208982349510001@barmar.ne.mediaone.net>
Newsgroups: misc.int-property
Lines: 52

In article <barmar-1208982349510001@barmar.ne.mediaone.net>, barmar@bbnplanet.com (Barry Margolin) wrote:

> In article <taquilla.1253324114C@news.erols.com>, taquilla@erols.com
> (Tracy Aquilla) wrote:
>
> >Not only does the claim avoid the word "algorithm", it is clearly directed
> >to an apparatus, not an algorithm.
>
> But if the only practical way to use the algorithm is by embodying it in
> an apparatus (a computer), what's the difference? All the lawyers seem to
> be making a distinction that doesn't exist in the real world, just so that
> they can justify the claim that you can't patent an algorithm.

Well, let me put this another way. Many anti-software advocates have excoriated software patents as improperly reaching to ideas and laws of nature. It is apparent and well-settled that what is going on in one's mind, and the learning and teaching of it shouldn't be protected.

It is true that nailing an algorithm to an apparatus may tie doesn all practical uses known at the time for some types of claims, but don't diminish the significance of those limitations. A number of recent internet commerce patents have just been neutered precisely by the court's narrow construction of those very limitations. The particular apparatus contemplated, even when the claim is recited as a "general purpose computer" may well be a significant limitation.

Then, to be patentable, understand that it is not only the manifestation of the algorithm on a machine that limits the patent, but also the particular application or class of application of the algorithm. See State Street Bank and the recent Microsoft case involving the patent directed to multithreading. Limitations to a particular application of multithreading (in that case, application to a text editor) were held to be the principal basis for patentability of the inventions, and the claims not so limited were held to be invalid for lack of written description.

So, its neither as good, nor as bad as you think it is.

> If you can
> patent a computer running a program that implements an algorithm, haven't
> you effectively patented the algorithm? As I said in another post, the
> only way it seems that someone could use the algorithm without infringing
> the patent would be on pencil and paper (or in one's head), which is not
> likely to be practical for any algorithm worth patenting.

But first you need to get the patent with such scope. Check out the multithreading case.

--
just another view,
Andy Greenberg
(remove "nonspam-" to get my real mailing address)

Subject: Re: XOR: 4,197,590 (was: Re: Patents and OSS)
Date: Tue, 04 Aug 1998 07:11:02 -0400
From: werdna@nonspam-gate.net (Andrew C. Greenberg)
Message-ID: <werdna-0408980711020001@tpafx1-56.gate.net>
References: <6q5dpb$l84$1@samba.rahul.net>
   <taquilla.1252560724A@news.erols.com>
Newsgroups: misc.int-property
Lines: 19

In article <6q5dpb$l84$1@samba.rahul.net>, c.c.eiftj@12.usenet.us.com (Rahul Dhesi) wrote:

> So you might ask: If the concept is so trivial, why didn't anybody else
> implement it before STAC?

snip of the argument made by Microsoft concerning validity of the STAC patent, which argument lost in Federal Court, leading to a $110M verdict against Microsoft.

> So the infamous STAC patent prevents others from doing what STAC did,
> because of the patent monopoly.

Which of the claims prevent people from reverse engineering MSDOS?

--
just another view,
Andy Greenberg
(remove "nonspam-" to get my real mailing address)

Subject: Re: XOR: 4,197,590 (was: Re: Patents and OSS)
Date: 4 Aug 1998 19:48:31 GMT
From: c.c.eiftj@12.usenet.us.com (Rahul Dhesi)
Message-ID: <6q7oif$5o8$1@samba.rahul.net>
References: <taquilla.1252618475A@news.erols.com>
<6q5dpb$l84$1@samba.rahul.net>
Newsgroups: misc.int-property
Lines: 27

In <taquilla.1252618475A@news.erols.com> taquilla@erols.com (Tracy
Aquilla) writes:

>>>>This is a matter of opinion. Why should anybody have a monopoly on a
>>>>concept as basic as "when compressing a device don't change its name"?
>>
>>>Which claim in the patent is directed to the concept of "when compressing a
>>>device don't change its name"?
>>
>>None of the claims is specifically directed to this concept. STAC could
>>not have successfully patented the concept as I have stated it.

>Then there is no monopoly on the concept. The claims define the monopoly.

If that really were so, Microsoft would have had no trouble achieving
the same goal (compress a device without changing its name) without
infringing the patent.

I wonder if you are reading solely the claims and ignoring the complete
specification. If you read the whole thing, it will become clear to you
that the purpose of the STAC patent was to achieve a monopoly on the
basic concept of "when compressing a device, don't change its name."
The claims were simply a means to that end.
--
Rahul Dhesi <dhesi@spams.r.us.com>
"Frankly, sir, nobody is that much of an expert on anything, at least
not on usenet." -- Andy Greenberg <werdna@gate.net>

Subject: Re: XOR: 4,197,590 (was: Re: Patents and OSS)
Date: Tue, 11 Aug 98 10:30:37 GMT
From: taquilla@erols.com (Tracy Aquilla)
Message-ID: <taquilla.1253219077A@news.erols.com>
References: <6qok5rjhd1@samba.rahul.net>
<6q4s7p$hv5$1@samba.rahul.net>
<werdna-0308980826590001@tpafx1-56.gate.net>
Newsgroups: misc.int-property
Lines: 59

In Article <6qok5rjhd1@samba.rahul.net>, c.c.eiftj@12.usenet.us.com (Rahul
Dhesi) wrote:
>In <werdna-1008982210200001@tstpa1-41.gate.net> werdna@nonspam-gate.net
>(Andrew C. Greenberg) writes:
>
>>The claims define the invention. You cannot claim an algorithm,
>>although the mentioning of an algorithm or formula as part of a claim does
>>not render the claim invalid.
>
>The argument has gone something like this:
>
> Rahul: <talks about algorithms protected by patents>
> Andy: Can't protect an algorithm with a patent.
> Rahul: Can. <gives examples>

No Rahul, you certainly have not given any examples. You mentioned a few
concepts, while claiming that they are protected by patents, but when asked
to identify a specific patent having a claim directed to a mere concept
(i.e. algorithm), you have been unable to identify either a specific patent
or a specific claim. (If are you referring to patent 4,197,590 as your
'example', which claim?)

> Andy: <oops, better change my wording> You cannot claim an
> algorithm, although the mentioning of an algorithm or
> formula as part of a claim does not render the claim
> invalid.
>
>Hmmm...so an algorithm can be incorporated in your claims...hmmm.....

Yes, but one cannot claim an algorithm. An algorithm is non-statutory
subject matter (see the MPEP).

>The fact is that algorithms do get patent protection in the USA.

Very loose use of the term 'fact' here (Rahul-speak perhaps?). Where is the
evidence? You say this nearly every week, but when challenged, you cannot
point to any patents that claim an algorithm.

>If you
>have a new algorithm on which you want a monopoly in the USA, you can
>get patent protection, provided it meets the usual criteria of novelty
>and nonobviousness.

One might be able to "get patent protection", but the claims cannot be
directed to an algorithm.

>You must word the claims in a certain way, and this

Yes, such that one does not claim the algorithm.

>I am not sure what to do.

Just cite a patent, and state which claim is directed to an algorithm.
Tracy

Thomas T. Aquilla, Ph.D.
Genetic Engineering and Biotech
Research and Consulting, Ltd.
taquilla@erols.com

Subject: Re: XOR: 4,197,590 (was: Re: Patents and OSS)
Date: Tue, 11 Aug 1998 23:57:03 -0700
From: Bruce Hayden <bhayden@ieee.org>
Message-ID: <35D13CBF.5DFEF1B6@ieee.org>
References: <barmar-1108982219050001@barmar.ne.mediaone.net>
<taquilla.1253219077A@news.erols.com>
Newsgroups: misc.int-property
Lines: 41

Barry Margolin wrote:
>
> In article <taquilla.1253219077A@news.erols.com>, taquilla@erols.com
> (Tracy Aquilla) wrote:
>
> >Yes, but one cannot claim an algorithm. An algorithm is non-statutory
> >subject matter (see the MPEP).

Well, that is not really true. After all, ALL method claims
are algorithms. What is prohibited are "mathematical" algorithms.

> It's true, you have to claim a computer running a program that implements
> an algorithm. But this is effectively equivalent to patenting the
> algorithm itself. The only other thing you might do with one of these
> algorithms is run them with pencil and paper, but they're generally
> useless when used that way, so the fact that the patent doesn't cover this
> use is not very interesting. So while you can't officially patent an
> algorithm, you can effectively do so.

Well, you have made the critical distinction here.
What you have to start with is the realization that a
general purpose computer running a particular program
is a special purpose computer, which is considered an
apparatus, and thus statutory.

Algorithms are, as pointed out above, methods or procedures, again
one of the statutory classes. Of course, you have to get around the
archaic Benson mathematical algorithm limitation. But that is easily
done by including sufficient hardware in the method claims. Thus,
if a method claim claims interaction between and/or among hardware
elements, then the claim is probably statutory.
--

The preceding was not a legal opinion, and is not my employer's.
Original portions Copyright 1998 Bruce E. Hayden,all rights reserved
My work may be copied in whole or part, with proper attribution,
as long as the copying is not for commercial gain.

Bruce E. Hayden bhayden@acm.org
Phoenix, Arizona bhayden@copatlaw.com
***Currently use the following: bhayden@uswest.net

Subject: Re: XOR: 4,197,590 (was: Re: Patents and OSS)
Date: Wed, 12 Aug 98 16:11:37 GMT
From: taquilla@erols.com (Tracy Aquilla)
Message-ID: <taquilla.1253325937D@news.erols.com>
References: <35D13CBF.5DFEF1B6@ieee.org>
Newsgroups: misc.int-property
Lines: 80

In Article <35D13CBF.5DFEF1B6@ieee.org>, Bruce Hayden <bhayden@ieee.org> wrote:
>Barry Margolin wrote:
>> In article <taquilla.1253219077A@news.erols.com>, taquilla@erols.com
>> (Tracy Aquilla) wrote:
>>
>> >Yes, but one cannot claim an algorithm. An algorithm is non-statutory
>> >subject matter (see the MPEP).
>
>Well, that is not really true. After all, ALL method claims
>are algorithms. What is prohibited are "mathematical" algorithms.

Sorry, I thought the discussion was regarding mathematical algorithms (or abstract mathematical concepts). Of course, methods are statutory, but an algorithm per se is not necessarily a statutory method.

>> It's true, you have to claim a computer running a program that implements
>> an algorithm.

Not necessarily.

>> But this is effectively equivalent to patenting the algorithm itself.

Not necessarily (see below).

>> The only other thing you might do with one of these
>> algorithms is run them with pencil and paper,

Or one could also possibly use a 'patented algorithm' in an improved or an entirely different method or apparatus. One cannot infringe an apparatus claim or a method claim by merely using an algorithm that is but a single element of the claim. Unless each and every limitation is met by the accused device or process, there can be no infringement, even if the algorithm is an element of the claim. Thus, the mere use of the algorithm itself is not enough to find infringement.

Unfortunately, until someone points to a real example (i.e. an issued patent with a claim directed to an algorithm), this discussion will remain abstract and unfocused.

>> but they're generally
>> useless when used that way, so the fact that the patent doesn't cover this
>> use is not very interesting.

It is quite interesting to me. There are probably many different ways to implement an algorithm, no?

>> So while you can't officially patent an algorithm, you can effectively do so.

Well we agree that one cannot "officially" patent an algorithm. Now, can you point to a claim in an issued patent which "effectively" does so?

>What you have to start with is the realization that a
>general purpose computer running a particular program
>is a special purpose computer, which is considered an
>apparatus, and thus statutory.

Of course, but an apparatus is not an algorithm. One cannot properly read the other limitations from the claim.

>Algorithms are, as pointed out above, methods or procedures, again
>one of the statutory classes. Of course, you have to get around the
>archaic Benson mathematical algorithm limitation. But that is easily
>done by including sufficient hardware in the method claims.

Thus, such additional limitations "effectively" prevent one from claiming the algorithm itself, because one cannot infringe a claimed apparatus without meeting each and every limitation of the claim (e.g. by employing an algorithm which is but a single limitation of the claim).

>if a method claim claims interaction between and/or among hardware
>elements, then the claim is probably statutory.

The question remains: does this prevent others from using the algorithm itself in other methods or apparatus? I do not believe that it does.
Tracy

Thomas T. Aquilla, Ph.D.
Genetic Engineering and Biotech
Research and Consulting, Ltd.
taquilla@erols.com

Subject: Re: XOR: 4,197,590 (was: Re: Patents and OSS)
Date: 12 Aug 1998 17:36:36 GMT
From: c.c.eiftj@12.usenet.us.com (Rahul Dhesi)
Message-ID: <6qsjr4$kv5$1@samba.rahul.net>
References: <taquilla.1253325937D@news.erols.com>
Newsgroups: misc.int-property
Lines: 18

In <taquilla.1253325937D@news.erols.com> taquilla@erols.com (Tracy Aquilla) writes:

>Well we agree that one cannot "officially" patent an algorithm. Now,
>can you point to a claim in an issued patent which "effectively" does
>so?

The broader claims of patent 4,558,302 claim the LZW algorithm for data comression, with some mathematical terms replaced with longer phrases, e.g., 'variable' changed to 'storage means', 'input string' changed to 'input signal stream', 'algorithm' changed to 'apparatus', etc.

LZW isn't just a proper subset of what is being claimed, LZW *is* what is being claimed, at least in the broadest claims.

--
Rahul Dhesi <dhesi@spams.r.us.com>
"Frankly, sir, nobody is that much of an expert on anything, at least
not on usenet." -- Andy Greenberg <werdna@gate.net>

Subject: Re: XOR: 4,197,590 (was: Re: Patents and OSS)
Date: Wed, 12 Aug 98 18:26:30 GMT
From: taquilla@erols.com (Tracy Aquilla)
Message-ID: <taquilla.1253334030A@news.erols.com>
References: <6qsjr4$kv5$1@samba.rahul.net>
Newsgroups: misc.int-property
Lines: 29

In Article <6qsjr4$kv5$1@samba.rahul.net>, c.c.eiftj@12.usenet.us.com (Rahul
Dhesi) wrote:
>In <taquilla.1253325937D@news.erols.com> taquilla@erols.com (Tracy
>Aquilla) writes:
>
>>Well we agree that one cannot "officially" patent an algorithm. Now,
>>can you point to a claim in an issued patent which "effectively" does
>>so?
>
>The broader claims of patent 4,558,302 claim the LZW algorithm for data
>comression, with some mathematical terms replaced with longer phrases,
>e.g., 'variable' changed to 'storage means', 'input string' changed to
>'input signal stream', 'algorithm' changed to 'apparatus', etc.

Where is "apparatus" defined as an algorithm? Nowhere I know of (including
the specification of the patent). An apparatus is not an algorithm.

>LZW isn't just a proper subset of what is being claimed, LZW *is* what
>is being claimed, at least in the broadest claims.

Well claim 1 is the broadest claim, and it is directed to an apparatus.
Perhaps you can explain the basis for your conclusion that the 'broader
claims' effectively claim the algorithm itself?
Tracy

Thomas T. Aquilla, Ph.D.
Genetic Engineering and Biotech
Research and Consulting, Ltd.
taquilla@erols.com

Subject: Re: XOR: 4,197,590 (was: Re: Patents and OSS)
Date: 12 Aug 1998 20:05:27 GMT
From: c.c.eiftj@12.usenet.us.com (Rahul Dhesi)
Message-ID: <6qssi7mji1@samba.rahul.net>
References: <taquilla.1253334030A@news.erols.com>
Newsgroups: misc.int-property
Lines: 20

In <taquilla.1253334030A@news.erols.com> taquilla@erols.com (Tracy
Aquilla) writes:

>>The broader claims of patent 4,558,302 claim the LZW algorithm for data
>>comression, with some mathematical terms replaced with longer phrases,
>>e.g., 'variable' changed to 'storage means', 'input string' changed to
>>'input signal stream', 'algorithm' changed to 'apparatus', etc.

>Where is "apparatus" defined as an algorithm? Nowhere I know of (including
>the specification of the patent). An apparatus is not an algorithm.

I don't care whether an apparatus is an algorithm or not.

The attorneys for Unisys, which inherited the patent from Univac, have
been quoted as saying that all implementations of LZW, whether in
hardware or software, are protected by the patent.
--
Rahul Dhesi <dhesi@spams.r.us.com>
"Frankly, sir, nobody is that much of an expert on anything, at least
not on usenet." -- Andy Greenberg <werdna@gate.net>

Subject: Re: XOR: 4,197,590 (was: Re: Patents and OSS)
Date: Thu, 13 Aug 1998 03:18:06 -0700
From: Bruce Hayden <bhayden@ieee.org>
Message-ID: <35D2BD5E.C007B14B@ieee.org>
References: <6qssi7mji1@samba.rahul.net>
Newsgroups: misc.int-property
Lines: 19

Rahul Dhesi wrote:

> I don't care whether an apparatus is an algorithm or not.
>
> The attorneys for Unisys, which inherited the patent from Univac, have
> been quoted as saying that all implementations of LZW, whether in
> hardware or software, are protected by the patent.

And what is wrong with that?
--

The preceding was not a legal opinion, and is not my employer's.
Original portions Copyright 1998 Bruce E. Hayden,all rights reserved
My work may be copied in whole or part, with proper attribution,
as long as the copying is not for commercial gain.

Bruce E. Hayden bhayden@acm.org
Phoenix, Arizona bhayden@copatlaw.com
***Currently use the following: bhayden@uswest.net

Subject: Re: XOR: 4,197,590 (was: Re: Patents and OSS)
Date: 18 Aug 1998 17:22:21 GMT
From: c.c.eiftj@12.usenet.us.com (Rahul Dhesi)
Message-ID: <6rcd8dsbo1@samba.rahul.net>
References: <35D987ED.3665B796@ieee.org>
<6rabvp$g5a$1@samba.rahul.net>
Newsgroups: misc.int-property
Lines: 49

In <35D987ED.3665B796@ieee.org> Bruce Hayden <bhayden@ieee.org> writes:

>Finally, when I indicate that software itself is not patentable, what
>is meant is that software in the abstract is not patentable. Thus,
>for example, in most cases, putting software in a book and publishing
>it will usually not result in infringement....

I have been in supermarkets where they had an aisle labelled 'software'. What I found there was paper plates and napkins. Obviously, I was not referring to such software in this discussion.

Almost equally obviously, I was not referring to 'program listings', which is the meaning with which you are using the word 'software' above. In fact I have never observed anybody use 'software' to refer merely to a printed listing. Usually people call it 'program listing' or 'source code' or 'source printout' or something similar.

'Software' here is an abbreviation for 'computer software', which refers in this particular context to programs that are in a form that may be loaded into a computer system and then executed. If you have a computer system with an optical scanner into which you can feed a program listing and have it immediately run, then perhaps a printed program listing should be considered software. This is not the case today but it could be so in the future.

Today, the average software buyer expects his software on a machine-readable medium that can be loaded and run.

Patent law allows any patent to be described on paper. In fact that is one of the purposes of patent law -- to encourage dissemination of ideas, and it's REQUIRED that a description of the patent on paper be made available. So far as I can tell, copies of patents (as published by the Patent Office) may be freely made and the making of such copies does not violate the patent. It's only a small step to say that paper copies of patented software will not violate the patent, unless such paper copies are functional in such a way that their existence could be considered an infringing 'use' of the patent.

Of course software in the abstract cannot be patented. Coat hangers in the abstract cannot be patented either. Machines in the abstract cannot be patented. NOTHING in the abstract can be patented. You are perfectly free to make as many drawings as you wish, and create as many images in your mind as you wish, of any patented invention. Software is not special in this respect.
--
Rahul Dhesi <dhesi@spams.r.us.com>
"And you say you're interested in the truth? What a joke....when noted authorities...contradict your view, you end up resorting to ad hominems....What a dodo." -- Ram Samudrala addressing Andrew C. Greenberg.

Subject: Re: XOR: 4,197,590 (was: Re: Patents and OSS)
Date: Wed, 19 Aug 1998 00:33:22 -0400
From: barmar@bbnplanet.com (Barry Margolin)
Message-ID: <barmar-1908980033220001@barmar.ne.mediaone.net>
References: <35D987ED.3665B796@ieee.org>
<6rabvp$g5a$1@samba.rahul.net>
Newsgroups: misc.int-property
Lines: 22

In article <35D987ED.3665B796@ieee.org>, Bruce Hayden <bhayden@ieee.org> wrote:

>While it is possible that all versions coded into a particular language
>that are loaded into RAM are patentable, what you have in your head
>is not, and is not contributorily infringing either.

It also doesn't do anything when it's in your head, so who cares?

Does it infringe when put on a floppy disk and mailed out to customers? That's certainly not an apparatus that actually performs the invention, but as soon as you stick it in a computer and tell it to run the program you have such an apparatus. When that happens, who is infringing -- the customer or the vendor? As a customer, should I have to verify that I don't turn my computer into an infringing device? And how am I supposed

to do so, since I am not privy to the mechanisms embodied in most of the software I run? This is presumably where contributory infringement comes into play.

--
Barry Margolin, barmar@bbnplanet.com
GTE Internetworking, Cambridge, MA
Support the anti-spam movement; see <http://www.cauce.org/>

Subject: Re: XOR: 4,197,590 (was: Re: Patents and OSS)
Date: Fri, 21 Aug 1998 07:38:26 -0400
From: werdna@nonspam-gate.net (Andrew C. Greenberg)
Message-ID: <werdna-2108980738260001@tstpa1-71.gate.net>
References: <35DBFCEA.E85A87CB@ieee.org>
<barmar-1908980033220001@barmar.ne.mediaone.net>
Newsgroups: misc.int-property
Lines: 19

In article <35DBFCEA.E85A87CB@ieee.org>, Bruce Hayden <bhayden@ieee.org> wrote:

> If you purchase software, load it into your computer, and that infringes
> a patent claim for a machine, then yes, you probably are the direct
> infringer. However, the company that provided the software to you
> is possibly a contributory infringer. Unfortunately, this means that
> you, or someone else in a similar situation, is probably going to be
> named as a defendant, along with the vendor, since you have to have
> a direct infringer in order to have a contributory infringer.

Since contributory infringement is an independently actionable tort, I doubt that a direct infringer needs to be named. I have seen pure contributory and inducement actions where only the derivatively liable parties are named. (This is also a great way to get good witnesses.)

--
just another view,
Andy Greenberg
(remove "nonspam-" to get my real mailing address)

Subject: Re: XOR: 4,197,590 (was: Re: Patents and OSS)
Date: Wed, 12 Aug 1998 22:26:25 -0400
From: werdna@nonspam-gate.net (Andrew C. Greenberg)
Message-ID: <werdna-1208982226250001@tstpa2-80.gate.net>
References: <6qsjr4$kv5$1@samba.rahul.net>
Newsgroups: misc.int-property
Lines: 33

In article <6qsjr4$kv5$1@samba.rahul.net>, c.c.eiftj@12.usenet.us.com
(Rahul Dhesi) wrote:

> The broader claims of patent 4,558,302 claim the LZW algorithm for data
> comression, with some mathematical terms replaced with longer phrases,
> e.g., 'variable' changed to 'storage means', 'input string' changed to
> 'input signal stream', 'algorithm' changed to 'apparatus', etc.

Which claim is Dhesi referring to? The broad claims of '302 are directed an apparatus.

> LZW isn't just a proper subset of what is being claimed, LZW *is* what
> is being claimed, at least in the broadest claims.

Dhesi cites no authority (or even a claim) in support of his proposition. Perhaps LZW is what the authors would have liked to have claimed, but it is far from clear what is the scope of the patent. Many software patents construed in recent Markman hearings have been narrowly limited to particular applications, despite the broad "algorithm-like" nature of their claim language. The E-Data patent is a nice example.

Yet another example is a recent patent asserted against Microsoft, alleged by anti-patent folks to be directed to all multithreading techniques. It was recently held invalid, not on the grounds of prior art, but on the ground that the specification narrowly required limitations not written into the patents, under the recent Gentry case.

These words, I do not think they mean what Rahul thinks they mean.

--
just another view,
Andy Greenberg
(remove "nonspam-" to get my real mailing address)

Subject: Re: XOR: 4,197,590 (was: Re: Patents and OSS)
Date: 13 Aug 1998 05:59:31 GMT
From: c.c.eiftj@12.usenet.us.com (Rahul Dhesi)
Message-ID: <6qtvc3spv1@samba.rahul.net>
References: <werdna-1208982226250001@tstpa2-80.gate.net>
Newsgroups: misc.int-property
Lines: 18

In <werdna-1208982226250001@tstpa2-80.gate.net> werdna@nonspam-gate.net
(Andrew C. Greenberg) writes:

>> LZW isn't just a proper subset of what is being claimed, LZW *is* what
>> is being claimed, at least in the broadest claims.

>Dhesi cites no authority (or even a claim) in support of his proposition.
>Perhaps LZW is what the authors would have liked to have claimed, but it
>is far from clear what is the scope of the patent....

Do you believe that there are certain implementations of LZW that are
not protected by the patent? If so, which implementations are these?
Do you know of any that exist, or are you simply guessing that some
will eventually be found?
--
Rahul Dhesi <dhesi@spams.r.us.com>
"Frankly, sir, nobody is that much of an expert on anything, at least
not on usenet." -- Andy Greenberg <werdna@gate.net>

Subject: Re: XOR: 4,197,590 (was: Re: Patents and OSS)
Date: Thu, 13 Aug 1998 07:51:48 -0400
From: werdna@nonspam-gate.net (Andrew C. Greenberg)
Message-ID: <werdna-1308980751480001@tstpa2-80.gate.net>
References: <6qtvc3spv1@samba.rahul.net>
Newsgroups: misc.int-property
Lines: 57

In article <6qtvc3spv1@samba.rahul.net>, c.c.eiftj@12.usenet.us.com
(Rahul Dhesi) wrote:

> In <werdna-1208982226250001@tstpa2-80.gate.net> werdna@nonspam-gate.net
> (Andrew C. Greenberg) writes:
>
> >> LZW isn't just a proper subset of what is being claimed, LZW *is* what
> >> is being claimed, at least in the broadest claims.
>
> >Dhesi cites no authority (or even a claim) in support of his proposition.
> >Perhaps LZW is what the authors would have liked to have claimed, but it
> >is far from clear what is the scope of the patent....
>
> Do you believe that there are certain implementations of LZW that are
> not protected by the patent? If so, which implementations are these?
> Do you know of any that exist, or are you simply guessing that some
> will eventually be found?

I'm not guessing at all. I'm stating that Mr. Dhesi hasn't made his case,
hasn't answered Tracy's questions and that his arguments so far manifest a
lack of understanding necessary to make that case. The mere statement
that "the broad claims cover everything" when the broad claim is
associated with an apparatus does not prove the "algorithm" is patented,
even if it weren't true. The fact that his sole response is to ask others
to prove negatives indicates his inability to make his proof.

Mr. Dhesi likes to gainsay well-established truths. It's what he does.
But on patent matters, Rahul is clueless. Please note that he has yet to
exemplify anything, despite his claims to the contrary and mere gainsay,
as Tracy noted, and Rahul was unable to refute.

The Supreme Court stated many times unequivocally that, "it is the claim
that defines the invention and gives notice to the public of the
limits of the patent monopoly," that quote coming from the recent
Warner-Jenkinson v. Hilton Davis case.

It is likewise well-established that claims wholly preempting the use of
an algorithm are unpatentable. The Court wrote that, "In Benson, we held
unpatentable claims for an algorithm used to convert binary code decimal
numbers to equivalent pure binary numbers. The sole practical application
of the algorithm was in connection with the programming of a general
purpose digital computer. We defined "algorithm" as a "procedure for
solving a given type of mathematical problem," and we concluded that such
an algorithm, or mathematical formula, is like a law of nature, which
cannot be the subject of a patent." That would be Diamond v. Diehr.

Here in the United States, which falls under the jurisdiction of the
Supreme Court, Rahul's ravings are legal lunacy, however rational they may
appear before the Court of Rahul. Until the Constitution is amended to
give legal credence to Rahul's lunacy, the gentle reader is invited to
consider for him or herself which view, the Supreme Court or Rahul's, more
accurately reflects what is the law.

--
just another view,
Andy Greenberg
(remove "nonspam-" to get my real mailing address)

Subject: Re: XOR: 4,197,590 (was: Re: Patents and OSS)
Date: 13 Aug 1998 19:09:09 GMT
From: c.c.eiftj@12.usenet.us.com (Rahul Dhesi)
Message-ID: <6qvdkl$85m$1@samba.rahul.net>
References: <werdna-1308980751480001@tstpa2-80.gate.net>
Newsgroups: misc.int-property
Lines: 75

The question is whether LZW is a proper subset of what is protected by

the LZW patent, or whether LZW itself is what is protected by the LZW patent. I believe LZW itself is protected, and Andy has been incessantly disagreeing. So I asked Andy:

> Do you believe that there are certain implementations of LZW that are not protected by the patent? If so, which implementations are these? Do you know of any that exist, or are you simply guessing that some will eventually be found?

This is a pretty reasonable question, I think. If LZW is not itself protected, then it should be possible to find implementations of LZW that do not infringe the patent.

In <werdna-1308980751480001@tstpa2-80.gate.net> werdna@nonspam-gate.net (Andrew C. Greenberg) writes:

>I'm not guessing at all. I'm stating that Mr. Dhesi hasn't made his case, >hasn't answered Tracy's questions and that his arguments so far manifest a >lack of understanding necessary to make that case. The mere statement >that "the broad claims cover everything" when the broad claim is >associated with an apparatus does not prove the "algorithm" is patented, >even if it weren't true. The fact that his sole response is to ask others >to prove negatives indicates his inability to make his proof.

Ok, quite a bit of name-calling here, but no example so far of any non-infringing implementation of LZW.

>Mr. Dhesi likes to gainsay well-established truths. It's what he does. >But on patent matters, Rahul is clueless. Please note that he has yet to >exemplify anything, despite his claims to the contrary and mere gainsay, >as Tracy noted, and Rahul was unable to refute.

Still loking for that counterexample. None found.

>The Supreme Court stated many times unequivocally that, "it is the claim >that defines the invention and gives notice to the public of the >limits of the patent monopoly," that quote coming from the recent >Warner-Jenkinson v. Hilton Davis case.

Argument by appeal to authority, but irrelevant authority in this case. No evidence that the Supreme Court actually determined that LZW is not protected.

>It is likewise well-established that claims wholly preempting the use of >an algorithm are unpatentable. The Court wrote that, "In Benson, we held >unpatentable claims for an algorithm used to convert binary code decimal >numbers to equivalent pure binary numbers. The sole practical application >of the algorithm was in connection with the programming of a general >purpose digital computer. We defined "algorithm" as a "procedure for >solving a given type of mathematical problem," and we concluded that such >an algorithm, or mathematical formula, is like a law of nature, which >cannot be the subject of a patent." That would be Diamond v. Diehr.

Apparently there is some other algorithm, which the Supreme Court determined to be not protected. But no sign that such a determination was made for LZW.

>Here in the United States, which falls under the jurisdiction of the >Supreme Court, Rahul's ravings are legal lunacy, however rational they may >appear before the Court of Rahul. Until the Constitution is amended to >give legal credence to Rahul's lunacy, the gentle reader is invited to >consider for him or herself which view, the Supreme Court or Rahul's, more >accurately reflects what is the law.

Oh my! If words could kill, I would be dead a hundred times over by now. Andy truly waxes eloquent here. I can only dream of using the English language as as fluently and colorfully as he does.

Now wouldn't it be nice if repeated name-calling could invalidate the LZW patent?
--
Rahul Dhesi <dhesi@spams.r.us.com>
"Frankly, sir, nobody is that much of an expert on anything, at least not on usenet." -- Andy Greenberg <werdna@gate.net>

Subject: Re: XOR: 4,197,590 (was: Re: Patents and OSS)
Date: Thu, 13 Aug 98 20:39:28 GMT
From: taquilla@erols.com (Tracy Aquilla)
Message-ID: <taquilla.1253428408A@news.erols.com>
References: <6qvdkl$85m$1@samba.rahul.net>
Newsgroups: misc.int-property
Lines: 80

In Article <6qvdkl$85m$1@samba.rahul.net>, c.c.eiftj@12.usenet.us.com (Rahul Dhesi) wrote:
>
> Do you believe that there are certain implementations of LZW that are
> not protected by the patent? If so, which implementations are these?

To be precise, nothing is actually "protected" by the patent. Patents do not "protect" an invention, they merely provide the patentee a legal right that others do not have - the right to file lawsuits for infringement of the patent. This is not protection, it is a remedy (or 'cause of action' in

legalese). If a patent actually "protected" an invention, there would be no need to sue, would there? (Unless of course, one gives an unusual meaning to the word "protection".)

More accurately, the question is whether any implementations of LZW that do not fall within the scope of the patent claims are known.

> Do you know of any that exist, or are you simply guessing that some
> will eventually be found? This is a pretty reasonable question, I think.

Yes, it is a reasonable question; a reasonable answer might be appropriately addressed by one of ordinary skill in the art (not me). Of course, another possibility is that 'unprotected implementations' are known, but not _publicly known_. However, a lack of such an example proves essentially nothing.

>If LZW is not itself protected, then it should be possible to find
>implementations of LZW that do not infringe the patent.

Simple logic dictates that this is not necessarily so; however, if it is possible to find implementations of LZW that do not infringe the patent, then clearly LZW is not itself protected (you just had it backwards).

>>The mere statement
>>that "the broad claims cover everything" when the broad claim is
>>associated with an apparatus does not prove the "algorithm" is patented,
>>even if it weren't true. The fact that his sole response is to ask others
>>to prove negatives indicates his inability to make his proof.
>
>Ok, quite a bit of name-calling here, but no example so far of any
>non-infringing implementation of LZW.

The lack of such examples is insufficient to prove your point. (BTW, where's the "name-calling"? Thankfully, I did not see any.)

>>It is likewise well-established that claims wholly preempting the use of
>>an algorithm are unpatentable. The Court wrote that, "In Benson, we held
>>unpatentable claims for an algorithm used to convert binary code decimal
>>numbers to equivalent pure binary numbers. The sole practical application
>>of the algorithm was in connection with the programming of a general
>>purpose digital computer. We defined "algorithm" as a "procedure for
>>solving a given type of mathematical problem," and we concluded that such
>>an algorithm, or mathematical formula, is like a law of nature, which
>>cannot be the subject of a patent." That would be Diamond v. Diehr.
>
>Apparently there is some other algorithm, which the Supreme Court
>determined to be not protected. But no sign that such a determination
>was made for LZW.

The law handed down in these cases applies to all algorithms. Which part of 'an algorithm cannot be the subject of a patent' do you not understand?

>Andy truly waxes eloquent here. I can only dream of using the
>English language as as fluently and colorfully as he does.

Agreed. Not only does he have a way with words, he makes logical arguments that are well supported - a noble adversary indeed.

>Now wouldn't it be nice if repeated name-calling could invalidate the
>LZW patent?

There was none, and anyway, who wants to invalidate the patent? The question was merely whether your assertion that "algorithms are protected by patents in the USA" is true or not. The validity of the patent is a different issue altogether.
Tracy

Thomas T. Aquilla, Ph.D.
Genetic Engineering and Biotech
Research and Consulting, Ltd.
taquilla@erols.com

Subject: Re: XOR: 4,197,590 (was: Re: Patents and OSS)
Date: Wed, 12 Aug 1998 12:02:24 -0400
From: Chris Pastel <crpastel@bpmlegal.com>
Message-ID: <35D1BC90.BEF3D35@bpmlegal.com>
References: <taquilla.1253219077A@news.erols.com>
Newsgroups: misc.int-property
Lines: 81

For once Rahul is using a word within its precise meaning. *Mathematical* algorithms cannot be patented, but *algorithms* are nothing more than a series of steps. In short, every method patent patents an algorithm, but not a mathematical algorithm. The standard court cases in this area (In re Allapat, etc.) make this distinction clear.

Tracy Aquilla wrote:

> In Article <6qok5rjhd1@samba.rahul.net>, c.c.eiftj@12.usenet.us.com (Rahul
> Dhesi) wrote:
> >In <werdna-1008982210200001@tstpa1-41.gate.net> werdna@nonspam-gate.net
> >(Andrew C. Greenberg) writes:
> >
> >>The claims define the invention. You cannot claim an algorithm,

> >>although the mentioning of an algorithm or formula as part of a claim does
> >>not render the claim invalid.
> >
> >The argument has gone something like this:
> >
> > Rahul: <talks about algorithms protected by patents>
> > Andy: Can't protect an algorithm with a patent.
> > Rahul: Can. <gives examples>
>
> No Rahul, you certainly have not given any examples. You mentioned a few
> concepts, while claiming that they are protected by patents, but when asked
> to identify a specific patent having a claim directed to a mere concept
> (i.e. algorithm), you have been unable to identify either a specific patent
> or a specific claim. (If are you referring to patent 4,197,590 as your
> 'example', which claim?)
>
> > Andy: <oops, better change my wording> You cannot claim an
> > algorithm, although the mentioning of an algorithm or
> > formula as part of a claim does not render the claim
> > invalid.
> >
> >Hmmm...so an algorithm can be incorporated in your claims...hmmm.....
>
> Yes, but one cannot claim an algorithm. An algorithm is non-statutory
> subject matter (see the MPEP).
>
> >The fact is that algorithms do get patent protection in the USA.
>
> Very loose use of the term 'fact' here (Rahul-speak perhaps?). Where is the
> evidence? You say this nearly every week, but when challenged, you cannot
> point to any patents that claim an algorithm.
>
> >If you
> >have a new algorithm on which you want a monopoly in the USA, you can
> >get patent protection, provided it meets the usual criteria of novelty
> >and nonobviousness.
>
> One might be able to "get patent protection", but the claims cannot be
> directed to an algorithm.
>
> >You must word the claims in a certain way, and this
>
> Yes, such that one does not claim the algorithm.
>
> >I am not sure what to do.
>
> Just cite a patent, and state which claim is directed to an algorithm.
> Tracy
>
> Thomas T. Aquilla, Ph.D.
> Genetic Engineering and Biotech
> Research and Consulting, Ltd.
> taquilla@erols.com

--
Christopher R. Pastel
Registered Patent Attorney
BROWN, PINNISI & MICHAELS, PC
118 North Tioga Street, Ste. 400
Ithaca, New York 14850
(607) 256-2000
Web Site: http://www.bpmlegal.com/
  or mirror site: http://www.lightlink.com/bbm

Subject: Re: XOR: 4,197,590 (was: Re: Patents and OSS)
Date: Wed, 12 Aug 1998 22:16:21 -0400
From: werdna@nonspam-gate.net (Andrew C. Greenberg)
Message-ID: <werdna-1208982216210001@tstpa2-80.gate.net>
References: <35D1BC90.BEF3D35@bpmlegal.com>
Newsgroups: misc.int-property
Lines: 26

In article <35D1BC90.BEF3D35@bpmlegal.com>, Chris Pastel
<crpastel@bpmlegal.com> wrote:

> For once Rahul is using a word within its precise meaning. *Mathematical*
> algorithms cannot be patented, but *algorithms* are nothing more than a series
> of steps. In short, every method patent patents an algorithm, but not a
> mathematical algorithm. The standard court cases in this area (In re Allapat,
> etc.) make this distinction clear.

It will be noted that I have used the term "legal algorithm" or similar
contexts to clarify that the notion of algorithm, as defined by the
Supreme Court *IS* narrower than the notion of an algorithm as Chris is
using it here. (They admitted as much in a footnote in Diamond v. Diehr,
noting that they did not pass upon the patentability or unpatentability of
subject matter falling in the broader notion of an algorithm.) The
definition they chose:

"We defined "algorithm" as a "procedure for solving a given type of mathematical problem," and we concluded that such an algorithm, or mathematical formula, is like a law of nature, which cannot be the subject of a patent."

--
just another view,
Andy Greenberg
(remove "nonspam-" to get my real mailing address)

Subject: Re: XOR: 4,197,590 (was: Re: Patents and OSS)
Date: Thu, 13 Aug 98 10:29:51 GMT
From: taquilla@erols.com (Tracy Aquilla)
Message-ID: <taquilla.1253391831B@news.erols.com>
References: <werdna-1208982216210001@tstpa2-80.gate.net>
Newsgroups: misc.int-property
Lines: 25

In Article <werdna-1208982216210001@tstpa2-80.gate.net>,
werdna@nonspam-gate.net (Andrew C. Greenberg) wrote:
>
>It will be noted that I have used the term "legal algorithm" or similar
>contexts to clarify that the notion of algorithm, as defined by the
>Supreme Court *IS* narrower than the notion of an algorithm as Chris is
>using it here. (They admitted as much in a footnote in Diamond v. Diehr,
>noting that they did not pass upon the patentability or unpatentability of
>subject matter falling in the broader notion of an algorithm.) The
>definition they chose:
>
> "We defined "algorithm" as a "procedure for solving a given type of
>mathematical problem," and we concluded that such an algorithm, or
>mathematical formula, is like a law of nature, which cannot be the
>subject of a patent."

This is almost identical to the dictionary definition I found in Webster's
II. Apparently the word "algorithm" is derived from the Greek word for
"number". Does anyone here know of an alternative definition found anywhere?
Tracy

Thomas T. Aquilla, Ph.D.
Genetic Engineering and Biotech
Research and Consulting, Ltd.
taquilla@erols.com

Subject: Re: XOR: 4,197,590 (was: Re: Patents and OSS)
Date: 13 Aug 1998 21:07:17 GMT
From: Dr. Ram Samudrala <ram.samudrala@stanford.nojunkemail>
Message-ID: <6qvki5$er9$1@nntp.Stanford.EDU>
References: <taquilla.1253391831B@news.erols.com>
Newsgroups: misc.int-property
Lines: 52

Tracy Aquilla <taquilla@erols.com> wrote:

>This is almost identical to the dictionary definition I found in
>Webster's II. Apparently the word "algorithm" is derived from the
>Greek word for "number". Does anyone here know of an alternative
>definition found anywhere?

From Algorithmics by David Harel (p x):

"An algorithm is an abstract recipe, prescribing a process that might
be carried out by a human, by a computer or by other means. It thus
represents a very general concept, with numerous applications."

Also Harel writes in p7:

"The word algorithm is derived from the name of the Persian
mathematician Mohammed al-Kowarizmi, who lived during the ninth
century, and who who is credited with providing the step-by-step rules
for adding, subtracting, multiplying and dividing ordinary decimal
numbers. When written in Latin, the name became Algorismus, from
which algorithm is a small step."

I recommend anyone seriously interested in algorithms to read Harel
for a light introduction... it's one of the most clearly written books
on the topic.

I hope your dictionary doesn't say it was derived from the Greek word
for number. This illustrates why a Merriam-Webster dictionary should
be used generally:

Main Entry: al7go7rithm
Pronunciation: 'al-g&-"ri-[th]&m
Function: noun
Etymology: alteration of Middle English algorisme, from Old French &
Medieval Latin; Old French, from Medieval Latin algorismus, from
Arabic al-khuwArizmi, from al-KhuwArizmi fl A.D. 825 Arabian
mathematician
: a procedure for solving a mathematical problem (as of finding the
greatest common divisor) in a finite number of steps that frequently
involves repetition of an operation; broadly : a step-by-step

procedure for solving a problem or accomplishing some end especially
by a computer
- al7go7rith7mic /"al-g&-'ri[th]-mik/ adjective
- al7go7rith7mi7cal7ly /-mi-k(&-)lE/ adverb

--Ram

email@urls || http://www.ram.org || http://www.twisted-helices.com/th
Based on the principle that if we were all crooks, we could at last be
uniform to some degree in the eyes of THE LAW... Once we had all broken
some kind of law, we'd all be in the same big happy club. ---Frank Zappa

---

Subject: Re: XOR: 4,197,590 (was: Re: Patents and OSS)
Date: Thu, 13 Aug 98 21:37:50 GMT
From: taquilla@erols.com (Tracy Aquilla)
Message-ID: <taquilla.1253431910B@news.erols.com>
References: <6qvki5$er9$1@nntp.Stanford.EDU>
Newsgroups: misc.int-property
Lines: 69

In Article <6qvki5$er9$1@nntp.Stanford.EDU>, Dr. Ram Samudrala
<ram.samudrala@stanford.nojunkemail> wrote:
>Tracy Aquilla <taquilla@erols.com> wrote:
>>In Article <werdna-1208982216210001@tstpa2-80.gate.net>,
>>werdna@nonspam-gate.net (Andrew C. Greenberg) wrote:
>>
>> Quoting the US Supreme Court:
>>> "We defined "algorithm" as a "procedure for solving a given type of
>>>mathematical problem," and we concluded that such an algorithm, or
>>>mathematical formula, is like a law of nature, which cannot be the
>>>subject of a patent."
>
>>This is almost identical to the dictionary definition I found in
>>Webster's II. Apparently the word "algorithm" is derived from the
>>Greek word for "number". Does anyone here know of an alternative
>>definition found anywhere?
>
>From Algorithmics by David Harel (p x):
>
>"An algorithm is an abstract recipe, prescribing a process that might
>be carried out by a human, by a computer or by other means. It thus
>represents a very general concept, with numerous applications."
>[snip]
>Merriam-Webster:
>a procedure for solving a mathematical problem (as of finding the greatest
>common divisor) in a finite number of steps that frequently
>involves repetition of an operation;

This is virtually identical to the definition in Webster's II, and the
Supremes' definition.

>broadly : a step-by-step procedure for solving a problem or accomplishing some
>end especially by a computer

The only difference is that Harel's definition appears to have eliminated
"mathematical", so it is a bit more broad, and Merriam-Webster specifically
notes the more broad, although qualified, use of the term. Note however that
Harel's definition of an algorithm as a "very general concept" and
"abstract" necessarily excludes patentable methods by definition.

>I hope your dictionary doesn't say it was derived from the Greek word
>for number.

Your chosen dictionary did not list the Greek source of the spelling of
algorithm? Webster's II says: Greek source _arithm_, "number".

>This illustrates why a Merriam-Webster dictionary should
>be used generally:

No, it simply illustrates that your chosen dictionary is incomplete (and mine).

The consensus is thus: an algorithm is a procedure for solving a
mathematical problem; an alternative, more broad definition includes
non-mathematical procedures or processes, especially as performed by a computer.

Apparently, the US law and the PTO use the more narrow definition of
algorithm to distinguish "algorithms" from "methods", because some methods
are patentable, while algorithms are expressly not patentable. Most
importantly, in context, a patentable "method" is not an "algorithm" under
US law, because at the PTO, algorithms are not patentable, while methods are.

Did we really need the Supreme Court, two dictionaries, and another book to
reach this conclusion?
Tracy

Thomas T. Aquilla, Ph.D.
Genetic Engineering and Biotech
Research and Consulting, Ltd.
taquilla@erols.com

Subject: Re: XOR: 4,197,590 (was: Re: Patents and OSS)
Date: 14 Aug 1998 02:09:49 GMT

From: Dr. Ram Samudrala <ram.samudrala@stanford.nojunkemail>
Message-ID: <6r069d$kl5$1@nntp.Stanford.EDU>
References: <taquilla.1253431910B@news.erols.com>
Newsgroups: misc.int-property
Lines: 24

Tracy Aquilla <taquilla@erols.com> wrote:

>Your chosen dictionary did not list the Greek source of the spelling
>of algorithm? Webster's II says: Greek source _arithm_, "number".

That's not what you wrote. You said: 'Apparently the word
"algorithm" is derived from the Greek word for "number".' The
etymology appears to be to the contrary.

I gave Harel's definition as an "alternative" to the one you posited.

I recommended using Merriam-Webster's because Webster's dictionaries
themselves may not be reliable (and this is the opinion of almost any
librarian I've spoken to on this matter).

I don't care whether algorithms are patentable or not; I ignore the
entire body of patent law when it comes to my actions.

--Ram

email@urls || http://www.ram.org || http://www.twisted-helices.com/th
"So it's going to cut down on technological innovations, efficiency
and so on, but it will happen to increase profits by accident.
Well, that's intellectual property rights." ---Noam Chomsky on GATT.

Subject: Re: XOR: 4,197,590 (was: Re: Patents and OSS)
Date: 14 Aug 1998 03:03:37 GMT
From: Dr. Ram Samudrala <ram.samudrala@stanford.nojunkemail>
Message-ID: <6r09e9lac1@nntp.Stanford.EDU>
References: <taquilla.1253447241F@news.erols.com>
<6r069d$kl5$1@nntp.Stanford.EDU>
Newsgroups: misc.int-property
Lines: 35

Tracy Aquilla <taquilla@erols.com> wrote:

>And apparently it is, at least in part.

Sorry, the only "authority" you've cited is a dictionary (and not a
Merriam-Webster one at that). I would appreciate a source to a text
on algorithms citing the etymology as you view it.

>Actually, it just appears to be more complex than you and I first
>thought.

I'm going by Harel's explanation on how the word came about. There's
nothing complex about that. And given that Harel's one of the
authorities on the subject, I'd like to a similar authority talk about
an etymology like the one you posited.

>Thank you.
[...]
>Thanks for the advice.

You're welcome.

>I already knew that - you so frequently mention it. :-)

That's just for the benefit of someone who may not be up to speed on
this group's dichotomy. I just didn't want anyone to think I had an
opinion about the patentability of algorithms in this thread.

--Ram

email@urls || http://www.ram.org || http://www.twisted-helices.com/th
There lived a certain man in Russia long ago,
He was big and strong in his eyes a flaming glow.
Most people looked at him with terror and with fear,
but to Moscow chicks he was such a lovely dear. ---Boney M, Rasputin

---

Subject: Re: XOR: 4,197,590 (was: Re: Patents and OSS)
Date: Fri, 14 Aug 1998 08:38:28 -0400
From: werdna@nonspam-gate.net (Andrew C. Greenberg)
Message-ID: <werdna-1408980838280001@tstpa1-20.gate.net>
References: <6r09e9lac1@nntp.Stanford.EDU>
Newsgroups: misc.int-property
Lines: 29

In article <6r09e9lac1@nntp.Stanford.EDU>, expt@alanine.ram.org wrote:

> Tracy Aquilla <taquilla@erols.com> wrote:
>
> >And apparently it is, at least in part.
>
> Sorry, the only "authority" you've cited is a dictionary (and not a

> Merriam-Webster one at that).

Ram is in serious credibility trouble here. My Webster's Third New International (which *IS* a Mirriam-Webster publication, by the way) likewise defines algorithm in terms of calculation and arithmetic. Indeed, solely so. Algorithm is defined in terms of the word algorism, which in turn is defined as the art of calculating with the symbols 0-9. A second definition of algorism is broader, "the art of calculating with any species of notation (the ~ of fractions)." Algorithm in this sense means the art of using a calculus or an algebra. [The last sentence was offered solely to complicate the matter :-)]

While I also prefer the M-W dictionaries, for completeness (they're just bigger) and sturdiness of binding, that has NOTHING to do with the matter at hand. The only relevant definition for purporses of patentability is the one given by the Supreme Court which, regardless of the views of librarians polled, is dispositive.

--
just another view,
Andy Greenberg
(remove "nonspam-" to get my real mailing address)

Subject: Re: XOR: 4,197,590 (was: Re: Patents and OSS)
Date: 15 Aug 1998 09:55:20 GMT
From: Dr. Ram Samudrala <ram.samudrala@stanford.nojunkemail>
Message-ID: <6r3lu8ibt1@nntp.Stanford.EDU>
References: <werdna-1508980021560001@tstpa1-34.gate.net>
<6r1srj$3dp$2@nntp.Stanford.EDU>
<werdna-1408980838280001@tstpa1-20.gate.net>
Newsgroups: misc.int-property
Lines: 27

Andrew C. Greenberg <werdna@nonspam-gate.net> wrote:

>OK, then I declare Tracy the winner. My fully Ram-compliant M-W Webster's
>Third New International reads as follows:

>al*go*rithm ..... n -s [alter. (influenced by artihmetic) of algorism]
>ALGORISM

>No other definitions are given. The etymological analysis in algorism, as
>noted beforehand, deals with arithmetic and numerals.

I trust David Harel more than your dictionary (if you do indeed have a Merriam-Webster's then the Merriam-Webster's dictionaries are contradicing each other, as the online one doesn't say anything about arithmetic and numerals in its etymological analysis).

You really honestly believe that the word algorithm didn't originate from the name of Mohammed al-Kowarizmi and instead believe it originates from the greek word for number (which is what Tracy is claiming)? Please reply clearly for posterity's sake here.

--Ram

email@urls || http://www.ram.org || http://www.twisted-helices.com/th
Based on the principle that if we were all crooks, we could at last be uniform to some degree in the eyes of THE LAW... Once we had all broken some kind of law, we'd all be in the same big happy club. ---Frank Zappa

---

Subject: Re: XOR: 4,197,590 (was: Re: Patents and OSS)
Date: Sat, 15 Aug 98 13:25:23 GMT
From: taquilla@erols.com (Tracy Aquilla)
Message-ID: <taquilla.1253575163B@news.erols.com>
References: <6r3lu8ibt1@nntp.Stanford.EDU>
Newsgroups: misc.int-property
Lines: 34

In Article <6r3lu8ibt1@nntp.Stanford.EDU>, Dr. Ram Samudrala
<ram.samudrala@stanford.nojunkemail> wrote:
>Andrew C. Greenberg <werdna@nonspam-gate.net> wrote:
>
>>OK, then I declare Tracy the winner. My fully Ram-compliant M-W Webster's
>>Third New International reads as follows:
>
>>al*go*rithm ..... n -s [alter. (influenced by artihmetic) of algorism]
>>ALGORISM
>
>>No other definitions are given. The etymological analysis in algorism, as
>>noted beforehand, deals with arithmetic and numerals.
>
>I trust David Harel more than your dictionary (if you do indeed have a
>Merriam-Webster's then the Merriam-Webster's dictionaries are
>contradicing each other,

Not uncommon - so goes your penulitmate source.

>You really honestly believe that the word algorithm didn't originate
>from the name of Mohammed al-Kowarizmi and instead believe it
>originates from the greek word for number. (which is what Tracy is
>claiming)?

Maybe the 'al' part did? The Greek word _arithm_ is simply the one with the closest spelling to "algorithm". Of course, every language probably has a word for number (doesn't your dictionary list several?), and many are similar.

>Please reply clearly for posterity's sake here.

It is both. Apparently the word "algorithm" was derived by combining part of a person's name with the concept of manipulating 'numbers'. Just this lay person's opinion though.
Tracy

Subject: Re: XOR: 4,197,590 (was: Re: Patents and OSS)
Date: Sat, 15 Aug 1998 11:21:59 -0400
From: werdna@nonspam-gate.net (Andrew C. Greenberg)
Message-ID: <werdna-1508981121590001@tstpa1-28.gate.net>
References: <6r3o1tiii1@nntp.Stanford.EDU>
<werdna-1508980021560001@tstpa1-34.gate.net>
Newsgroups: misc.int-property
Lines: 73

In article <6r3o1tiii1@nntp.Stanford.EDU>, expt@alanine.ram.org wrote:

> And just in case it's not completely clear (by my words in the
> previous posts in this thread, which it should be unless people aren't
> reading the previous posts), I consider David Harel (or any or
> algorithmican of his calibre) to be the authorative source.'

I wonder if Ram's prior interrogation of librarians were also polled on their views of David Harel? As I recall, Rahul thought the world of M-W dictionaries until one contradicted him.

With all due respect, does Ram truly believe that someone who writes a book about algorithms from a computer science perspective has any deeper comprehension or understanding of the etymology of that word than a lexicographer? I am unaware that David's C.V. shows any expertise as a historian, and references in his bibliography to chapter 1 are solely C.S. references, so far as I can tell. [I confess not spending a whole lot of time on this silliness -- and it is silly.]

My beef is simply this. Ram didn't argue the issue, or provide further authority. He simply MADE UP a poll of librarians to gainsay one reference while lording yet another reference above it. I didn't lionize M-W dictionaries as the final word on matters. He did.

When faced with a contrdiction between the publisher's flagship publication and his desired result, he decided to argue beased on David Harel's authority AS A HISTORIAN AND ETYMOLOGIST!!!

I have no doubt that the word algorithm derives from the word "algorism," which refers to arabic numbers, and which in turn did derive from the persian with the name I do not dare attempting to spell without opening another book (which I simply will not do at this point). [By the way, the fact that the word was named after a persion, does not mean the word was coined in the persian langauge. I am no expert in the etymology of the word, not do I pretend any insight into various references -- I certainly don't conduct (or concoct) librarian polls to settle disputes -- I'm just amazed that Ram would hold himself out as an authority to mediate between authorities]

I just wanted the audience to see how deeply Ram thought this out before he began spouting off at the mouth about references.

Since this is aprapos of nothing -- I shall continue to use my Webster's Third New International as an exceptional lexicon, occasionally refer to my New Lexicon Webster's, notwithstanding Ram's survey evidence as a backup reference and look to historians for my historical references.

As to the legal meaning of "mathematical algorithm," I shall defer to the Supreme Court of the United States rather than the Court of Ram, and shall not quibble with the Supreme Court's reference to the term algorithm in its mathematical context, particularly given the etymology and dictionary definitions of the term.

I suggest that those seriously concerned about the Supreme Court's ability to comprehend the oridinary and technical meaning of words consider the footnote in Diamond v. Diehr, which acknowledges broader uses of the term and discusses those meanings in conext of the opinion as a whole.

And as to argumentum poll-the-librarians, followed by computer-scientist-ergo-historian: phooey! I don't care whether Ram arrived thereby at the right result or not -- his reasoning is the worst form of ad hoc demagoguery, chosing references to suit the result he desires and then bad-mouting one over another; and, indeed, the furtherst from intellectually honest truth-seeking we have seen for quite a while.

For my part, I'd like to know the truth of things -- I know upon whom I shall rely in the future -- and whose assertions of carefully considered views I shall discount.

--
just another view,
Andy Greenberg
(remove "nonspam-" to get my real mailing address)

Subject: Re: XOR: 4,197,590 (was: Re: Patents and OSS)
Date: Thu, 13 Aug 1998 23:45:10 -0700
From: Bruce Hayden <bhayden@ieee.org>
Message-ID: <35D3DCF6.25838445@ieee.org>
References: <taquilla.1253431910B@news.erols.com>
Newsgroups: misc.int-property
Lines: 33

Tracy Aquilla wrote:
>
> The consensus is thus: an algorithm is a procedure for solving a
> mathematical problem; an alternative, more broad definition includes
> non-mathematical procedures or processes, especially as performed by a computer.

Actually, I would suggest that that is not the consensus, esp. as to
patent law. There is a distinction between "mathematical" algorithms,
and all others. The former are not patentable, the later often are.

> Apparently, the US law and the PTO use the more narrow definition of
> algorithm to distinguish "algorithms" from "methods", because some methods
> are patentable, while algorithms are expressly not patentable. Most
> importantly, in context, a patentable "method" is not an "algorithm" under
> US law, because at the PTO, algorithms are not patentable, while methods are.

Again, it is "mathematical" algorithms that are not patentable.

> Did we really need the Supreme Court, two dictionaries, and another book to
> reach this conclusion?
> Tracy

Given that you weren't quite correct, then yes.
--

The preceding was not a legal opinion, and is not my employer's.
Original portions Copyright 1998 Bruce E. Hayden,all rights reserved
My work may be copied in whole or part, with proper attribution,
as long as the copying is not for commercial gain.

Bruce E. Hayden bhayden@acm.org
Phoenix, Arizona bhayden@copatlaw.com
***Currently use the following: bhayden@uswest.net

Subject: Re: XOR: 4,197,590 (was: Re: Patents and OSS)
Date: Fri, 14 Aug 98 13:11:17 GMT
From: taquilla@erols.com (Tracy Aquilla)
Message-ID: <taquilla.1253487917A@news.erols.com>
References: <35D3DCF6.25838445@ieee.org>
Newsgroups: misc.int-property
Lines: 43

In Article <35D3DCF6.25838445@ieee.org>, Bruce Hayden <bhayden@ieee.org> wrote:
>Tracy Aquilla wrote:
>>
>> The consensus is thus: an algorithm is a procedure for solving a
>> mathematical problem; an alternative, more broad definition includes
>> non-mathematical procedures or processes, especially as performed by a
computer.
>
>Actually, I would suggest that that is not the consensus, esp. as to
>patent law.

Sorry for the confusion, I should have been more precise - it is the
consensus of the authorities that were cited in this thread.

>There is a distinction between "mathematical" algorithms,
>and all others. The former are not patentable, the later often are.

Yes, I already stated that patentable "methods" are distinguished from
(mathematical) "algorithms". If algorithms are often patentable, perhaps you
can point to an issued patent wherein the claimed invention is an algorithm
(non-mathematical, of course)?

>> Apparently, the US law and the PTO use the more narrow definition of
>> algorithm to distinguish "algorithms" from "methods", because some methods
>> are patentable, while algorithms are expressly not patentable. Most
>> importantly, in context, a patentable "method" is not an "algorithm" under
>> US law, because at the PTO, algorithms are not patentable, while methods are.
>
>Again, it is "mathematical" algorithms that are not patentable.

I agree - "mathematical" algorithms are not patentable.

>> Did we really need the Supreme Court, two dictionaries, and another book to
>> reach this conclusion?
>> Tracy
>
>Given that you weren't quite correct, then yes.

Weren't quite correct about what? Do you know of an issued patent wherein
the claimed invention is a non-mathematical "algorithm" (i.e. not a device
or method under 101)? I am unfamiliar with the art in this area, but I am

interested in learning.
Tracy

Subject: Re: XOR: 4,197,590 (was: Re: Patents and OSS)
Date: Thu, 13 Aug 1998 23:56:51 -0700
From: Bruce Hayden <bhayden@ieee.org>
Message-ID: <35D3DFB3.B910944B@ieee.org>
References: <6qvmo2$9rk$1@samba.rahul.net>
<6qvki5$er9$1@nntp.Stanford.EDU>
Newsgroups: misc.int-property
Lines: 48

Rahul Dhesi wrote:

> The second meaning makes every precisely-stated algorithm a mathematical
> algorithm. The first meaning makes every algorithm involving
> mathematics a mathematical algorithm. Since all computer processing
> involves mathematical techniques, at the very least involving
> comparisons and logical operations, again all algorithms implemented on
> computer systems must be mathematical algorithms. Thus it is clearly
> incorrect to say that 'mathematical algorithms' do not get patent
> protection.

You are entitled to define the term any way that you want.
However, that has nothing to do with what is or is not patentable.
The term "mathematical" algorithm in patent law has been signficantly
limited to an algorithm where you have a handful of numbers input,
and one or a small number of numbers output. Under this definition
treating a computer program as if it were a mathematical algorithm
is insufficient to take it out of the statutory realm.

> Anybody arguing that 'mathematical algorithms' don't get patent
> protection needs to properly describe what he means by 'mathematical
> algorithm' in such a way that known patented algorithms that take an
> abstract data stream as input (such as IDEA and RSA encrytion and LZW
> compression) are not included in this meaning.

I like Andy's distinction of patentable algorithms and nonpatentable
algorithms. In any case, the big thing that these algorithms
(in whatever guise) have is that they relate very strongly to
hardware. There must be hardware involved. Software without
hardware is still pretty much unpatentable. However, loading it
in RAM usually creates a statutory machine. Writing it to a
floppy or a CD-ROM usually creates a statutory article of manufacture.
Specifying a structural relationship between hardware elements
in method claims usually results in a statutory process.

But my point here is that getting tied up in exactly what is a
"mathematical" algorithm in the common vernacular, and what is not
is totally irrelevant in determining what is patentable subject matter.
--

The preceding was not a legal opinion, and is not my employer's.
Original portions Copyright 1998 Bruce E. Hayden,all rights reserved
My work may be copied in whole or part, with proper attribution,
as long as the copying is not for commercial gain.

Bruce E. Hayden bhayden@acm.org
Phoenix, Arizona bhayden@copatlaw.com
***Currently use the following: bhayden@uswest.net

Subject: Re: XOR: 4,197,590 (was: Re: Patents and OSS)
Date: Fri, 14 Aug 1998 07:57:56 -0400
From: werdna@nonspam-gate.net (Andrew C. Greenberg)
Message-ID: <werdna-1408980757560001@tstpa1-20.gate.net>
References: <6qvmo2$9rk$1@samba.rahul.net>
<6qvki5$er9$1@nntp.Stanford.EDU>
Newsgroups: misc.int-property
Lines: 16

In article <6qvmo2$9rk$1@samba.rahul.net>, c.c.eiftj@12.usenet.us.com
(Rahul Dhesi) wrote:

> Anybody arguing that 'mathematical algorithms' don't get patent
> protection needs to properly describe what he means by 'mathematical
> algorithm' in such a way that known patented algorithms that take an
> abstract data stream as input (such as IDEA and RSA encrytion and LZW
> compression) are not included in this meaning.

Asked, and answered, several times with direct quotes from pertinent
Supreme Court authority.

--
just another view,
Andy Greenberg
(remove "nonspam-" to get my real mailing address)

Subject: Re: XOR: 4,197,590 (was: Re: Patents and OSS)
Date: Fri, 14 Aug 1998 10:42:41 -0400
From: Chris Pastel <crpastel@bpmlegal.com>

Message-ID: <35D44CE1.4A853770@bpmlegal.com>
References: <6qvmo2$9rk$1@samba.rahul.net>
<6qvki5$er9$1@nntp.Stanford.EDU>
Newsgroups: misc.int-property
Lines: 69

Rahul Dhesi wrote:

> <snip>

> The second meaning makes every precisely-stated algorithm a mathematical
> algorithm. The first meaning makes every algorithm involving
> mathematics a mathematical algorithm. Since all computer processing
> involves mathematical techniques, at the very least involving
> comparisons and logical operations, again all algorithms implemented on
> computer systems must be mathematical algorithms. Thus it is clearly
> incorrect to say that 'mathematical algorithms' do not get patent
> protection.
>

Why do you persist in saying that 'mathematical algorithims' get patent protection? Perhaps you believe that the phrase 'mathematical algorithm' as used by the courts covers more than the courts say? Or perhaps you believe that 'patent protection' is something it is not? Andy has correctly stated the rules for determining infringement of a patent. Educate yourself about infringement and about the current law on patenting mathematical algorithms and you won't be persisting in such absurd statements.

Perhaps an example will help:

I claim:
Apparatus for measuring power consumption of a logic device, comprising:

means for determining an input slew rate of an input signal to said logic device;

means, responsive to said slew rate, for calculating a first power consumption resulting from a through-current of said logic device and including means for applying a formula

$$Wp=b+mt,$$

where

Wp is said power consumption,
t is said slew rate, and
b and m are constants;

means for calculating a second power consumption resulting from a charging current of said logic device; and

means for summing said first power consumption and said second power consumption to provide a total power consumption of said logic device.

This is patentable (5,473,548). According to your logic, the mathematical algorithm of Wp=b+mt is protected, whereas the equation itself is most assuredly NOT protected. Anyone is free to use the equation, they just can't use this particular apparatus. This is not a trivial distinction. As Andy nicely puts it, you can't patent the law of gravity, but you can patent the pendulum that uses it.

Side note: a lot of the discussion boils down to definitions. If you are going to use a word in a legal discussion or argument, then you should abide by the 'legal' meaning of the word and not get wound around the axle by insisting on the 'plain' or 'everyday' meaning.
--
Christopher R. Pastel
Registered Patent Attorney
BROWN, PINNISI & MICHAELS, PC
118 North Tioga Street, Ste. 400
Ithaca, New York 14850
(607) 256-2000
Web Site: http://www.bpmlegal.com/
or mirror site: http://www.lightlink.com/bbm

Subject: Re: XOR: 4,197,590 (was: Re: Patents and OSS)
Date: Sun, 16 Aug 1998 11:54:21 -0700
From: Bruce Hayden <bhayden@ieee.org>
Message-ID: <35D72ADD.22613BBD@ieee.org>
References: <6r1q18$nh8$1@samba.rahul.net>
<35D44CE1.4A853770@bpmlegal.com>
Newsgroups: misc.int-property
Lines: 105

Rahul Dhesi wrote:

> >Why do you persist in saying that 'mathematical algorithims' get patent
> >protection? Perhaps you believe that the phrase 'mathematical
> >algorithm' as used by the courts covers more than the courts say?
>
> I asked for a good definition of 'mathematical algorithm' and all I got
> was supercilious responses telling me to look at some court case. That

> doesn't cut it. If you are going to use a term in a manner not
> consistent with its plain English meaning, you need to be able to
> present a useful definition.

Sorry, that is not the way that the law works. Rather, the term has developed what we might call judicial gloss. And that is precisely what happens - you have to look at the supporting caselaw to determine exactly what a legal term means, and thanks to Benson, et al., the term "mathematical algorithm" has developed a judicial meaning.

>
> Talking about 'mathematical algorithm' "as used by the courts" boils
> down to defining 'mathematical algorithm' in terms of whether or not it
> gets patent protection. This is a useless and tautological definition
> of the term.

No, you just approach it too simplistically. First, you are correct that we define it almost tautologically - if something is patentable, then it isn't a mathematical algorithm. But we are forced into it because of the Supreme Court cases. Thus, we are forced to define anything that is patentable as not a mathematical algorithm.

You ask, What is going on here? The answer is that you are seeing how the law grows over time. The Supreme Court is the final court determining the law of the United States. Thus, everything they do at some level is cast into stone. But circumstances change over time. The way that the law adapts to this in many cases is that the old law is "distinguished". What this really means in many cases is that some term or another is interpreted more narrowly, allowing the old case to mean something else.

The Supreme Court can of course just overrule itself in order to change direction. The lower courts cannot. However note that even this is in serious dispute in the Supreme Court, with J. Souter leading the one it is law, it is cast into stone forever crowd, and J. Scalia leading the crowd willing to overrule itself. This difference of judicial opinion is one reason that these two justices seem to fight more in their opinions than others.

So, you have what I consider to have been a stupid decision, written by one of the most anti-patent justices of modern times (J. Douglas), that is totally out of touch with reality. So, what do you do? The Supreme Court started the ball rolling in Diamond v. Diehr, where they pointed at the term "mathematical algorithm" and pointed out that they obviously didn't have such in their case, rather, they had a machine, which was just as obviously patentable (using the English definition for the word "obvious" rather than the patent law definition).

So, ever since (and probably actually before) Diehr, the lower courts, most particularly the Federal Circuit that has exclusive jurisdiction over patent appeals, have been using this technique to provide patentability of ever more software.

Thus, you really do have a situation where the term "mathematical algorithm" is being driven by what is patentable, since if it is patentable, it cannot legally be a "mathematical algorithm". And of course what is patentable is to some extent driven by the lower court cases. Andy would like the Supreme Court to reopen this area of the law, and thinks that maybe State Street Bank may be a good test case. But until they do, the Federal Circuit in concert with the PTO determines what is patentable, and thus what is not a "mathematical algorithm".

That said, the question then is "what is patenable"? To some extent, the PTO's "Examination Guidelines for Computer-Related Inventions" that can be downloaded from:
http://www.uspto.gov/web/offices/com/hearings/software/analysis/computer.html
is fairly helpful. It includes a flowchart that does a decent job of following the Federal Circuit caselaw as of about 1/1/96. However, I have always felt the Exmaination Guidelines to be somewhat conservative compared to what you can actually get through the PTO.

> This sort of talking down seems to be typical of most of the attorneys
> posting here.
>
> I'm still waiting for somebody to provide an example of an
> implementation of LZW that is not protected by the LZW patent. The best
> disproof to an assertion of 'all A is B' is to find a B that is not A,
> and this Andy et al have consistently failed to do in the case of LZW.
> I see repeated claims that LZW is not protected by patent, but
> absolutely no evidence that any implementation of LZW exists, that is
> not covered by the patent.

Well, you won't get that from me, since I find it irrelevant whether all implementations of LZW are covered by the patent or not. That has absolutely nothing to do with whether the patent is valid or not.

--

The preceding was not a legal opinion, and is not my employer's.
Original portions Copyright 1998 Bruce E. Hayden,all rights reserved
My work may be copied in whole or part, with proper attribution,
as long as the copying is not for commercial gain.

Bruce E. Hayden bhayden@acm.org

Phoenix, Arizona bhayden@copatlaw.com
***Currently use the following: bhayden@uswest.net

Subject: Re: XOR: 4,197,590 (was: Re: Patents and OSS)
Date: Mon, 17 Aug 1998 21:06:51 -0700
From: Bruce Hayden <bhayden@ieee.org>
Message-ID: <35D8FDDB.54362F6A@ieee.org>
References: <6raabdfqo1@samba.rahul.net>
<35D72ADD.22613BBD@ieee.org>
Newsgroups: misc.int-property
Lines: 56

Rahul Dhesi wrote:
>
> Which again says that J-mathematical-algorithm has no intended plain
> English meaning, and in fact does not even have a clear judicial
> meaning; it's simply a label we use to refer to those algorithms that
> have been determined to be not patentable.

> What's the point of telling me that 'mathematical algorithm' has indeed
> been defined and I should look at such-and-such court case? I think
> it's very clear that this term has no definition. It's just a label
> used to refer to certain things, much as 'Bruce' and 'Andy' are labels
> used to refer to certain people. The term itself has no inherent
> meaning.

Rather its definition is determined by the software patent cases.
What you seem to want is a one or two line definition. You aren't
going to get one. That is because the term is now a legal term of art.
And legal terms of art invariably cannot be defined that cleanly.

What you have to keep in mind is that the term does have a meaning.
It is just that the term stands for the string of court cases.

> Let me propose that any time a term is used on Usenet for which a
> judicial meaning is intended which is substantially different from its
> plain English meaning, a J- prefix be used to signify this. This would
> make arguments a bit more clear:
>
> A: Algorithms can be patented!
> B: You're wrong, J-algorithms can't be patented!
> A: You could be right, perhaps J-algorithms can't be patented. But
> algorithms can be patented.
> B: Ok, we are both right. Let's shake hands.
> A: What on earth is a J-algorithm, by the way?
> B: Gee, I'm not sure. The Supreme Court isn't very clear about
> this.
> A: Ah, my condolences!

At this point, I would suggest talking to a patent attorney or agent.

> My assertion was that all implementations of LZW are protected by the
> patent. Anybody disagreeing with me really ought to find an
> implementation that is not protected by patent.

I might be willing to agree to a statement that all practical
impelementations of LZW with today's technology are protected by
the patent.
--

The preceding was not a legal opinion, and is not my employer's.
Original portions Copyright 1998 Bruce E. Hayden,all rights reserved
My work may be copied in whole or part, with proper attribution,
as long as the copying is not for commercial gain.

Bruce E. Hayden bhayden@acm.org
Phoenix, Arizona bhayden@copatlaw.com
***Currently use the following: bhayden@uswest.net

Subject: Re: XOR: 4,197,590 (was: Re: Patents and OSS)
Date: Wed, 12 Aug 1998 22:21:05 -0400
From: werdna@nonspam-gate.net (Andrew C. Greenberg)
Message-ID: <werdna-1208982221050001@tstpa2-80.gate.net>
References: <6qsh13kej1@samba.rahul.net>
<35D1BC90.BEF3D35@bpmlegal.com>
Newsgroups: misc.int-property
Lines: 30

In article <6qsh13kej1@samba.rahul.net>, c.c.eiftj@12.usenet.us.com
(Rahul Dhesi) wrote:

> Can the alleged distinction between 'algorithm' and 'mathematical
> algorithm' be summarized here in plain English? This is not a
> distinction made by either computer scientists or mathematicians.

I agree, there isn't a meaningful distinction for computer scientists or
mathematicians. The Supreme Court articulated THEIR definition of the
specialized legal term of art "algorithm" as follows:

"We defined "algorithm" as a "procedure for solving a given type of
mathematical problem," and we concluded that such an algorithm, or
mathematical formula, is like a law of nature, which cannot be the

subject of a patent."

They acknowledged this is narrower than more general and traditional notions of "algorithm," which is why they chose the "mathematical" qualifier. It is THEIR nomenclature, which has regrettably led to great confusion later on. (You should see what they did with the word "malice" in Times v. Sullivan!)

> BTW, for obvious reasons, I object to your use of the phrase 'for once'.

Me, too. I didn't think Dhesi was precise that time either. :-)

--
just another view,
Andy Greenberg
(remove "nonspam-" to get my real mailing address)

Subject: Re: XOR: 4,197,590 (was: Re: Patents and OSS)
Date: Thu, 13 Aug 1998 00:18:21 -0400
From: barmar@bbnplanet.com (Barry Margolin)
Message-ID: <barmar-1308980018210001@barmar.ne.mediaone.net>
References: <werdna-1208982221050001@tstpa2-80.gate.net>
Newsgroups: misc.int-property
Lines: 15

In article <werdna-1208982221050001@tstpa2-80.gate.net>,
werdna@nonspam-gate.net (Andrew C. Greenberg) wrote:

>I agree, there isn't a meaningful distinction for computer scientists or
>mathematicians. The Supreme Court articulated THEIR definition of the
>specialized legal term of art "algorithm" as follows:

So the whole reason for the argument in this thread is a stupid
terminology distinction? Can we lay it to rest and agree that it *is*
possible to patent a computer algorithm?

--
Barry Margolin, barmar@bbnplanet.com
GTE Internetworking, Cambridge, MA
Support the anti-spam movement; see <http://www.cauce.org/>

Subject: Re: XOR: 4,197,590 (was: Re: Patents and OSS)
Date: Thu, 13 Aug 98 10:32:30 GMT
From: taquilla@erols.com (Tracy Aquilla)
Message-ID: <taquilla.1253391990E@news.erols.com>
References: <barmar-1308980018210001@barmar.ne.mediaone.net>
Newsgroups: misc.int-property
Lines: 26

In Article <barmar-1308980018210001@barmar.ne.mediaone.net>,
barmar@bbnplanet.com (Barry Margolin) wrote:
>In article <werdna-1208982221050001@tstpa2-80.gate.net>,
>werdna@nonspam-gate.net (Andrew C. Greenberg) wrote:
>
>>I agree, there isn't a meaningful distinction for computer scientists or
>>mathematicians. The Supreme Court articulated THEIR definition of the
>>specialized legal term of art "algorithm" as follows:
>
>So the whole reason for the argument in this thread is a stupid
>terminology distinction?

So far, I have seen only two definitions cited that are supported by any
authority, and they are both the same. Where can one find the alternative
definition of "algorithm" to which you refer?

>Can we lay it to rest and agree that it *is*
>possible to patent a computer algorithm?

Without any convincing evidence? Certainly not. ;-)
Tracy

Thomas T. Aquilla, Ph.D.
Genetic Engineering and Biotech
Research and Consulting, Ltd.
taquilla@erols.com

Subject: Re: XOR: 4,197,590 (was: Re: Patents and OSS)
Date: Thu, 13 Aug 1998 08:02:40 -0400
From: werdna@nonspam-gate.net (Andrew C. Greenberg)
Message-ID: <werdna-1308980802400001@tstpa2-80.gate.net>
References: <barmar-1308980018210001@barmar.ne.mediaone.net>
Newsgroups: misc.int-property
Lines: 13

In article <barmar-1308980018210001@barmar.ne.mediaone.net>,
barmar@bbnplanet.com (Barry Margolin) wrote:

> So the whole reason for the argument in this thread is a stupid
> terminology distinction? Can we lay it to rest and agree that it *is*
> possible to patent a computer algorithm?

No.

--
just another view,
Andy Greenberg
(remove "nonspam-" to get my real mailing address)

Subject: Re: XOR: 4,197,590 (was: Re: Patents and OSS)
Date: Thu, 13 Aug 98 10:31:24 GMT
From: taquilla@erols.com (Tracy Aquilla)
Message-ID: <taquilla.1253391924D@news.erols.com>
References: <barmar-1208982352200001@barmar.ne.mediaone.net>
<6q4s7p$hv5$1@samba.rahul.net>
<werdna-0308980826590001@tpafx1-56.gate.net>
Newsgroups: misc.int-property
Lines: 22

In Article <barmar-1208982352200001@barmar.ne.mediaone.net>,
barmar@bbnplanet.com (Barry Margolin) wrote:
>In article <taquilla.1253334030A@news.erols.com>, taquilla@erols.com
>(Tracy Aquilla) wrote:
>
>>>LZW isn't just a proper subset of what is being claimed, LZW *is* what
>>>is being claimed, at least in the broadest claims.
>>
>>Well claim 1 is the broadest claim, and it is directed to an apparatus.
>>Perhaps you can explain the basis for your conclusion that the 'broader
>>claims' effectively claim the algorithm itself?
>
>Perhaps you can explain how someone could usefully perform the LZW
>algorithm without using such an apparatus?

By using an apparatus that does not meet all of the limitations of the claim.
Tracy

Thomas T. Aquilla, Ph.D.
Genetic Engineering and Biotech
Research and Consulting, Ltd.
taquilla@erols.com

Subject: Re: XOR: 4,197,590 (was: Re: Patents and OSS)
Date: Thu, 13 Aug 98 22:29:25 GMT
From: taquilla@erols.com (Tracy Aquilla)
Message-ID: <taquilla.1253435005D@news.erols.com>
References: <6qvmo2$9rk$1@samba.rahul.net>
<6q4s7p$hv5$1@samba.rahul.net>
<werdna-0308980826590001@tpafx1-56.gate.net>
Newsgroups: misc.int-property
Lines: 62

In Article <6qvmo2$9rk$1@samba.rahul.net>, c.c.eiftj@12.usenet.us.com (Rahul
Dhesi) wrote:
>
>The word algorithm has been used quite correctly here to refer to any
>step-by-step recipe. It's also true that the patent system in the USA
>recognizes algorithms as statutory subject matter, referring to them as
>'methods' or 'processes'.

The "the patent system in the USA" distinguishes between "methods", which
are patentable, and "algorithms", which are not patentable. See MPEP 2106
and 35 USC section 101.

"If the steps of gathering and substituting values were alone sufficient, every
mathematical equation, formula, or algorithm having any practical use would
be per se subject to patenting as a "process" under 101." MPEP 2106.

Certainly some "methods" are patentable, but if that is the only evidence
you have to offer showing that "algorithms" are patentable, your argument is
absolutely unconvincing.

> 1 : of, relating to, or according with mathematics
> 2 a : rigorously exact : PRECISE b : CERTAIN
> 3 : possible but highly improbable <only a mathematical chance>
>
>The second meaning makes every precisely-stated algorithm a mathematical
>algorithm.

Well, according to Ram's source, algorithms are "abstract", and a "very
general concept", which is the opposite of "precise" or "exact", no?

>Thus it is clearly incorrect to say that 'mathematical algorithms' do not get
>patent protection.

"A process that merely manipulates an abstract idea or performs a purely
mathematical algorithm is nonstatutory. In Sarkar, 588 F.2d at 1335, 200
USPQ at 139." MPEP 2106.

"For such subject matter to be statutory, the claimed process must be
limited to a practical application of the abstract idea or mathematical
algorithm in the technological arts. See Alappat, 33 F.3d at 1543, 31 USPQ2d
at 1556-57 (quoting Diamond v. Diehr, 450 U.S. at 192, 209 USPQ at 10)."

MPEP 2106.

>Anybody arguing that 'mathematical algorithms' don't get patent
>protection needs to properly describe what he means by 'mathematical
>algorithm' in such a way that known patented algorithms that take an
>abstract data stream as input (such as IDEA and RSA encrytion and LZW
>compression) are not included in this meaning.

"In practical terms, claims define nonstatutory processes if they:
- consist solely of mathematical operations without some claimed practical application (i.e., executing a "mathematical algorithm"); or

- simply manipulate abstract ideas, e.g., a bid (Schrader, 22 F.3d at 293-94, 30 USPQ2d at 1458-59) or a bubble hierarchy (Warmerdam, 33 F.3d at 1360, 31 USPQ2d at 1759), without some claimed practical application." MPEP 2106.

Thomas T. Aquilla, Ph.D.
Genetic Engineering and Biotech
Research and Consulting, Ltd.
taquilla@erols.com

Subject: Re: XOR: 4,197,590 (was: Re: Patents and OSS)
Date: 14 Aug 1998 01:00:09 GMT
From: c.c.eiftj@12.usenet.us.com (Rahul Dhesi)
Message-ID: <6r026pcss1@samba.rahul.net>
References: <taquilla.1253435005D@news.erols.com>
Newsgroups: misc.int-property
Lines: 36

In <taquilla.1253435005D@news.erols.com> taquilla@erols.com (Tracy Aquilla) writes:

[numerous quotes from MPEP]

[numerous other quotes]

The quotes you present cerainly sound impressive. However, none of them says that algorithms cannot get patent protection. They do make some assertions using vague terms that you do not define. For example:

>"For such subject matter to be statutory, the claimed process must be
>limited to a practical application of the abstract idea or mathematical
>algorithm in the technological arts.

What's a 'practical application' and what's a 'technological art'?
Suppose I discovered two new and nonobvious functions Fa and Fb, such that

Fa(Sa) = Sb

Fb(Sb) = Sa

where Sa is an arbitrary string and Sb is a significantly shorter string. Would you consider this a practical application that could get patent protection? If not, why not?

It seems to me that any algorithm that fulfills a need must be considered to be of practical use and hence eligible for patent protection, if it fulfils other requirements such as being new and nonobvious.
--
Rahul Dhesi <dhesi@spams.r.us.com>
"Frankly, sir, nobody is that much of an expert on anything, at least not on usenet." -- Andy Greenberg <werdna@gate.net>

Subject: Re: XOR: 4,197,590 (was: Re: Patents and OSS)
Date: Fri, 14 Aug 98 01:41:44 GMT
From: taquilla@erols.com (Tracy Aquilla)
Message-ID: <taquilla.1253446544E@news.erols.com>
References: <6r026pcss1@samba.rahul.net>
Newsgroups: misc.int-property
Lines: 28

In Article <6r026pcss1@samba.rahul.net>, c.c.eiftj@12.usenet.us.com (Rahul Dhesi) wrote:
>In <taquilla.1253435005D@news.erols.com> taquilla@erols.com (Tracy
>Aquilla) writes:
>
>[numerous quotes from MPEP]
>
>[numerous other quotes]
>
>The quotes you present cerainly sound impressive. However, none of them
>says that algorithms cannot get patent protection. They do make some
>assertions using vague terms that you do not define. For example:
>
>>"For such subject matter to be statutory, the claimed process must be
>>limited to a practical application of the abstract idea or mathematical
>>algorithm in the technological arts.
>

>What's a 'practical application' and what's a 'technological art'?

The MPEP is replete with specific examples, and I already cited the appropriate sections for you. If you read it, you might even find some rational arguments to support your position. ;-)
Tracy

Thomas T. Aquilla, Ph.D.
Genetic Engineering and Biotech
Research and Consulting, Ltd.
taquilla@erols.com

Subject: Re: XOR: 4,197,590 (was: Re: Patents and OSS)
Date: Fri, 14 Aug 1998 08:21:12 -0400
From: werdna@nonspam-gate.net (Andrew C. Greenberg)
Message-ID: <werdna-1408980821120001@tstpa1-20.gate.net>
References: <6r026pcss1@samba.rahul.net>
Newsgroups: misc.int-property
Lines: 66

In article <6r026pcss1@samba.rahul.net>, c.c.eiftj@12.usenet.us.com
(Rahul Dhesi) wrote:

> In <taquilla.1253435005D@news.erols.com> taquilla@erols.com (Tracy
> Aquilla) writes:
>
> [numerous quotes from MPEP]
>
> [numerous other quotes]
>
> The quotes you present cerainly sound impressive. However, none of them
> says that algorithms cannot get patent protection.

Dhesi in denial. He needs to read them again.

> They do make some
> assertions using vague terms that you do not define. For example:
>
> >"For such subject matter to be statutory, the claimed process must be
> >limited to a practical application of the abstract idea or mathematical
> >algorithm in the technological arts.
>
> What's a 'practical application' and what's a 'technological art'?

He needs to read them again. These terms are very well-understood and
defined in the references.

> Suppose I discovered two new and nonobvious functions Fa and Fb, such
> that
>
> Fa(Sa) = Sb
>
> Fb(Sb) = Sa
>
> where Sa is an arbitrary string and Sb is a significantly shorter
> string. Would you consider this a practical application that could get
> patent protection? If not, why not?

The abstract function descibed in this manner is expressly unpatentable.
Something quite like it is patentable, but only in precisely the same
sense you can get a patent on the law of gravity (which you can't). The
pendulum, exploiting the law of gravity in a particular way, is
patentable. A device that exploits the law of gravity in a different way
is noninfringing. A device for shortening disk files, exploiting the
functions Fa and Fb in a particular way, is patentable. A device for
shortening disk files exploiting the functions in a different way is
noninfringing.

> It seems to me that any algorithm that fulfills a need must be
> considered to be of practical use and hence eligible for patent
> protection, if it fulfils other requirements such as being new and
> nonobvious.

Dhesi reverts to wordplay. The Supreme Court said he is wrong, as recited
in previous postings, so Dhesi is now trying to use the same words to mean
different things, hopefully evolving his previously unsupported assertions
of fact to a more supportable position. Happily, he is finally coming
around to the points that tracy, Bruce and I made at the beginning of this
thread: abstract algorithms are not patentable per se, but particular
implementations can be so patented, and the scope of those inventions are
determined, indeed DEFINED by, the claims of the patent.

--
just another view,
Andy Greenberg
(remove "nonspam-" to get my real mailing address)

Subject: Re: XOR: 4,197,590 (was: Re: Patents and OSS)
Date: 14 Aug 1998 14:26:27 GMT
From: galibert@pobox.com (Olivier Galibert)
Message-ID: <slrn6t8i8f.3o7.galibert@renaissance.loria.fr>
References: <werdna-1408980821120001@tstpa1-20.gate.net>

Newsgroups: misc.int-property
Lines: 24

In article <werdna-1408980821120001@tstpa1-20.gate.net>, Andrew C. Greenberg wrote:
>> It seems to me that any algorithm that fulfills a need must be
>> considered to be of practical use and hence eligible for patent
>> protection, if it fulfils other requirements such as being new and
>> nonobvious.
>
>Dhesi reverts to wordplay. The Supreme Court said he is wrong, as recited
>in previous postings, so Dhesi is now trying to use the same words to mean
>different things, hopefully evolving his previously unsupported assertions
>of fact to a more supportable position. Happily, he is finally coming
>around to the points that tracy, Bruce and I made at the beginning of this
>thread: abstract algorithms are not patentable per se, but particular
>implementations can be so patented, and the scope of those inventions are
>determined, indeed DEFINED by, the claims of the patent.

Please explain me that, since for the so-called LZW patent nobody in the whole OSS community has been able to come up with a program implementing an algorithm giving the same output for the same set of inputs which wouldn't infringe at least the first claim, pretending that the LZW algorithm itself isn't in fact protected is anything but wordplay?

OG.

Subject: Re: XOR: 4,197,590 (was: Re: Patents and OSS)
Date: Sat, 15 Aug 98 13:18:19 GMT
From: taquilla@erols.com (Tracy Aquilla)
Message-ID: <taquilla.1253574739A@news.erols.com>
References: <6r2ojfdo1@samba.rahul.net>
<6qok5rjhd1@samba.rahul.net>
Newsgroups: misc.int-property
Lines: 44

In Article <6r2ojfdo1@samba.rahul.net>, c.c.eiftj@12.usenet.us.com (Rahul Dhesi) wrote:
>In <barmar-1308982057360001@barmar.ne.mediaone.net> barmar@bbnplanet.com
>(Barry Margolin) writes:
>>In article <taquilla.1253428408A@news.erols.com>, taquilla@erols.com
>>(Tracy Aquilla) wrote:
>
>>>To be precise, nothing is actually "protected" by the patent. Patents
>>>do not "protect" an invention, they merely provide the patentee a
>>>legal right that others do not have - the right to file lawsuits for
>>>infringement of the patent.
>
>>A patent protects the inventor's (actually, the patent holder's) right
>>to control use of the invention.

Patents do not necessarily give the patentee the 'right to control' either (the right to exclude is not an exclusive right). Often a patentee must obtain a license to practice his own invention.

>>But I think you understand what we
>>lay people mean when we abbreviate this as "protects the invention".

Yes, I believe I do. I simply re-stated the issue more precisely. "Protection" is rather vague. I was trying to focus on how 'protected by a patent' is actually defined under the law, because the scope of "protection" is rather narrow. The precise question is: is there an embodiment of the algorithm that does not fall _within the scope of the claims_?

>Nope, there is no misunderstanding.
>
>This is simply another example of when we use the plain and obvious
>meaning of some word,

What do you think is the plain and ordinary meaning of "protect"? If a patent really protects an invention, then why are there lawsuits for infringement?

>Patents do 'protect' inventions, and federal law says so.
>
>From 35 USC 41(2):

Yes, Congress too sometimes uses words somewhat imprecisely. You may consult your Merriam-Webster dictionary for the plain, ordinary meaning. :-)
Tracy

Subject: Re: XOR: 4,197,590 (was: Re: Patents and OSS)
Date: Sat, 15 Aug 98 13:27:37 GMT
From: taquilla@erols.com (Tracy Aquilla)
Message-ID: <taquilla.1253575297C@news.erols.com>
References: <6r3o1tiii1@nntp.Stanford.EDU>
<6q4s7p$hv5$1@samba.rahul.net>
<werdna-0308980826590001@tpafx1-56.gate.net>
Newsgroups: misc.int-property
Lines: 20

In Article <6r3o1tiii1@nntp.Stanford.EDU>, Dr. Ram Samudrala
<ram.samudrala@stanford.nojunkemail> wrote:
>
>I consider David Harel (or any or
>algorithmican of his calibre) to be the authorative source.

I disagree. He might be the most authoritative source for the common definition as used in the technical arts, but I believe a standard dictionary of the English language is the more appropriate source for the plain, ordinary meaning, as used by the public in general. Anyway, now we have consulted several dictionaries, the US Supreme Court, and David Harel.

>Harel's the expert here.

Is he a renowned professor of the English language?

>Please reply clearly for posterity's sake here.

See my previous post.
Tracy

Terry Ritter, his current address, and his top page.

Last updated: 1999-02-20

Patents and Personal Use

A *Ciphers By Ritter* Page

What are the limits of anything and doing a patented thing? Can we make for our own use? Can we give to others?

This is a forum for the art of cryptography and the crossfield of topics. We invite message posts on the topics of this forum to be in conformance with our rules.

Contents

- 2006-03-28 Patents Rant - How does about to claim, and so the inventor, has an will, a
- 2006-03-28 Re: Patents Rant - Re: Patents Rant - Re: Patents Rant - Re: Patents Rant - Re:
- 2006-03-28 Re: Patents Rant - This is what trying. Science - Tried, Force Work - practical and
- 2006-03-28 Re: Patents Rant - Re: Patents Rant - Re: Patents Rant - Re: Patents Rant - Re:
- 2006-03-28 Re: Patents Rant - Re: Patents Rant - Re: Patents Rant - Re: Patents Rant - Re:
- 2006-03-28 Re: Patents Rant - There is no "correct" response to the question.
- 2006-03-28 Re: Patents Rant - Re: Patents Rant - Re: Patents Rant - Re: Patents Rant -

patented invention, it would be impractical to sue that individual for
infringement.

But it may be that the patentee is looking for somebody to make an example
of, regardless of the cost of litigation ...

Lee Hollaar

Just published in AIPLA Quarterly Journal -- how Congress should change the
patent laws to address software patents. See http://www.cs.utah.edu/~hollaar

---

Subject: Re: Patent infringement (was: "Experts slam PGP encryption")
Date: 26 Mar 1997 21:03:21 GMT
From: hollaar@ursa0.cs.utah.edu (Lee Hollaar)
Message-ID: <5hc2up$i11@magus.cs.utah.edu>
References: <33392CB4.3E5E@worldnet.att.net>
<5hb4pf$cem@magus.cs.utah.edu>
Newsgroups:
alt.security.pgp,comp.security.pgp.discuss,alt.politics.org.nsa,sci.crypt,talk.politics.crypto
Lines: 59

In article <33392CB4.3E5E@worldnet.att.net> emDOTpeaDOTsea@worldnet.att.net
writes:
>Lee Hollaar wrote:
: Let's see what the statute says --
:
: 35 U.S.C. 271. Infringement of patent
: (a) Except as otherwise provided in this title, whoever without
: authority makes, uses, offers to sell, or sells any patented invention, within
: the United States or imports into the United States any patented invention
: during the term of the patent therefor, infringes the patent.
: (b) Whoever actively induces infringement of a patent shall be liable
: as an infringer.
: (c) Whoever offers to sell, or sells within the United States or
: imports into the United States a component of a patented machine, manufacture,
: combination or composition, or a material or apparatus for use in practicing a
: patented process, constituting a material part of the invention, knowing the
: same to be especially made or especially adapted for use in an infringement of
: such patent, and not a staple article or commodity of commerce suitable for
: substantial noninfringing use, shall be liable as a contributory infringer.
: * * *
:
: Infringement for "offers to sell" and "imports" was added, effective January
: 1, 1996, by the GATT implementation act.
:
: The original United States patent statute, the Patent Act of 1790 passed by
: the first Congress, provided similar rights to the patentee -- "the sole and
: exclusive right and liberty of making, constructing, using and vending to
: others to be used, the said invention or discovery".
:
: So, clearly an individual making or using a patented invention is an infringer
: of the patent, except as limited by the narrow "experimental use" exception
: "for the sole purpose of gratifying a philosophical taste, or curiosity, or
: for mere amusement". See _Roche Products v. Bolar Pharmaceutical_, 733 F2d
: 858, 221 USPQ 937 (Fed. Cir. 1984).

>Ahhh .... if we're gonna get legalistic, better consider rewording your
>interpretation (above); nothing wrong with *using* a patented
>invention. Did you mean " ... making or using *a copy of* a patented
>invention ...CRYPHTML.HTM"?

It's hard not to "get legalistic" when we are talking about what a law
permits or requires.

The reason it is not an infringement to use a patented invention if the
invention was originally purchased from the patent owner or a licensee is
the "patent exhaustion doctrine" which says that a patent owner gets the
full measure of his or her patent rights when they sell the thing covered
by the patent, so there is an implied license to use along with the sale.

If you are a user of a patent invention not made by permission of the
patent owner, then you are an infringer because the patent exhaustion
doctrine does not apply.

You don't talk about copies of a patented invention -- that's copyright
language. You do talk about something that infringes the patent, or
something that the patent claims "read on". If the patented invention
is claimed as a process, anything that performs the process infringes
the patent, whether it is a copy of the patentee's implementation or
not.

Lee Hollaar

Subject: Re: Patent infringement (was: "Experts slam PGP encryption")
Date: 29 Mar 1997 18:43:28 GMT
From: hollaar@ursa0.cs.utah.edu (Lee Hollaar)
Message-ID: <5hjnsg$pou@magus.cs.utah.edu>
References: <5helc5pfj1@trojan.neta.com>
<5hc2up$i11@magus.cs.utah.edu>
Newsgroups:
alt.security.pgp,comp.security.pgp.discuss,alt.politics.org.nsa,sci.crypt,talk.politics.crypto
Lines: 21

In article <5helc5pfj1@trojan.neta.com> blair@trojan.neta.com (Blair P Houghton)
writes:
>I thought all patents were essentially patents of
>processes, and the description of a device was a necessary
>(and sometimes sufficient) part of describing the process
>the device performs.

No, you can get a on "any new and useful process, machine, manufacture,
or composition of matter, or any new and useful improvement thereof"
(35 USC 101). A manufacture is just some man-made thing that may not
be a machine. And you don't have to indicate which of the four classes
your invention falls into.

Sometimes an invention will fall into two or more classes, and will be
separately claimed in each class. For example, a software-based
invention could be claimed as a process, a machine performing the
process, or even a manufacture of a floppy disk containing the software.

Besides utility patents, the United States grants patents on asexually
reproduced plants and ornamental designs, but these are far less common.

Lee Hollaar

Subject: Re: "Experts slam PGP encryption"
Date: 23 Mar 1997 12:57:56 GMT
From: hollaar@ursa0.cs.utah.edu (Lee Hollaar)
Message-ID: <5h39ck$f70@magus.cs.utah.edu>
References: <3334E7CB.1696@sternlight.com>
<1997Mar2302.29.06.7044@koobera.math.uic.edu>
Newsgroups:
alt.security.pgp,comp.security.pgp.discuss,alt.politics.org.nsa,talk.politics.crypto
Lines: 26

In article <3334E7CB.1696@sternlight.com> david@sternlight.com writes:
>D. J. Bernstein wrote:
>> David Sternlight <david@sternlight.com> wrote:
>> > Your comment is an interpretation of your own and is not consistent with
>> > the law,
>>
>> Nonsense. There is a firmly established experimental use exception in
>> the case law. There was an article in High Tech L. J. in 1992 discussing
>> the current boundaries of the doctrine.
>
>What are "the current boundaries of the doctrine"? And what might a more
>specific pointer to the article be?

The leading case on the experimental use defense is _Roche Products v.
Bolar Pharmaceutical_, 733 F.2d 858, 221 USPQ 937 (Fed. Cir. 1984).
(Given that, any law librarian can direct you to the case.)

In that case, the court said that the only uses that fall within the
exception are those made "for amusement, to satisfy idle curiosity
or for strictly philosophical inquiry."

It's a pretty narrow defense.

Lee Hollaar
Professor of Computer Science
Registered Patent Agent

Subject: Re: "Experts slam PGP encryption"
Date: 23 Mar 1997 17:54:37 GMT
From: djb@koobera.math.uic.edu (D. J. Bernstein)
Message-ID: <1997Mar2317.54.37.8375@koobera.math.uic.edu>
References: <5h39ck$f70@magus.cs.utah.edu>
Newsgroups:
alt.security.pgp,comp.security.pgp.discuss,alt.politics.org.nsa,talk.politics.crypto
Lines: 12

Lee Hollaar <hollaar@ursa0.cs.utah.edu> wrote:
> In that case, the court said that the only uses that fall within the
> exception are those made "for amusement, to satisfy idle curiosity
> or for strictly philosophical inquiry."

False. There are two allowed forms of experimental use. The first is
``to ascertain the verity and exactness of the specification.'' The
second, which _Roche_ called ``truly narrow'' and defined as you said,
is ``philosophical experimentation.''

---Dan
Let your users manage their own mailing lists. http://pobox.com/~djb/qmail.html

---

Subject: Re: "Experts slam PGP encryption"
Date: Sun, 23 Mar 1997 10:40:23 -0800
From: David Sternlight <david@sternlight.com>
Message-ID: <33357917.1FE9@sternlight.com>
References: <1997Mar2317.54.37.8375@koobera.math.uic.edu>
Newsgroups:
alt.security.pgp,comp.security.pgp.discuss,alt.politics.org.nsa,talk.politics.crypto
Lines: 36

D. J. Bernstein wrote:
>
> Lee Hollaar <hollaar@ursa0.cs.utah.edu> wrote:
> > In that case, the court said that the only uses that fall within the
> > exception are those made "for amusement, to satisfy idle curiosity
> > or for strictly philosophical inquiry."
>
> False.

Perhaps you could, with profit, learn the difference between "false" and
"incomplete". What he said was true but incomplete.

If you wish to nit-pick, you might argue that he used the word "only"
which is literally false according to you, but the spirit of the
discussion as well as what follows "only" mean the substantive clause is
true, but there's more. We wouldn't want to mislead, would we?

>There are two allowed forms of experimental use. The first is
> ``to ascertain the verity and exactness of the specification.'' The
> second, which _Roche_ called ``truly narrow'' and defined as you said,
> is ``philosophical experimentation.''
>

In either case, it's clear that neither of your descriptions covers Phil
Zimmermann's giving a copy of software practicing the patent, nor does
it cover giving a copy away for wide distribution (which was what this
part of the discussion was about).

What is more, I'd argue that his programming of PGP itself went well
beyond either definition, since he stated his intent in a letter to Jim
Bidzos, and by applying for a license. His intent was clearly more than
experimentation, and he's on record to that effect. Once he was on
record that his intent was dissemination, on being denied a license I
assert he was obligated to destroy, not distribute PGP.

David

Subject: Re: "Experts slam PGP encryption"
Date: 25 Mar 1997 01:21:43 GMT
From: hollaar@ursa0.cs.utah.edu (Lee Hollaar)
Message-ID: <5h79b7$c6j@magus.cs.utah.edu>
References: <33357917.1FE9@sternlight.com>
Newsgroups:
alt.security.pgp,comp.security.pgp.discuss,alt.politics.org.nsa,talk.politics.crypto
Lines: 33

In article <33357917.1FE9@sternlight.com> david@sternlight.com writes:
Somehow, the original Bernstein article did not make it to my machine,
so I'll go with the sternlight response ...

>D. J. Bernstein wrote:
>>
>> Lee Hollaar <hollaar@ursa0.cs.utah.edu> wrote:
>> > In that case, the court said that the only uses that fall within the
>> > exception are those made "for amusement, to satisfy idle curiosity
>> > or for strictly philosophical inquiry."
>>
>> False.
>>There are two allowed forms of experimental use. The first is
>> ``to ascertain the verity and exactness of the specification.'' The
>> second, which _Roche_ called ``truly narrow'' and defined as you said,
>> is ``philosophical experimentation.''

Here is the actual quote from _Roche Products v. Bolar Pharmaceutical_:
[B]y 1861, the law was "well-settled that an experiment with a
patented article for the sole purpose of gratifying a philosophical
taste, or curiosity, or for mere amusement is not an infringement
of the rights of the patentee." Peppenhausen v. Falke, 19 F. Cas.
1048, 1049 (C.C.S.D.N.Y. 1861)

So the major difference between what I said and what the Federal Circuit
quoted with approval is that I talked about the "only uses" and they
said "sole purpose".

As to "to acertain the verity and exactness of the specification," a quick
search of USPQ (from Volume 80 to present) doesn't find that quote. Could
you please furnish a cite, particularly to a Federal Circuit case or one
adopted by them?

Lee

Subject: Re: "Experts slam PGP encryption"
Date: 25 Mar 1997 03:30:53 GMT
From: djb@koobera.math.uic.edu (D. J. Bernstein)
Message-ID: <1997Mar2503.30.53.14543@koobera.math.uic.edu>
References: <5h79b7$c6j@magus.cs.utah.edu>
Newsgroups:
alt.security.pgp,comp.security.pgp.discuss,alt.politics.org.nsa,talk.politics.crypto
Lines: 21

Lee Hollaar <hollaar@ursa0.cs.utah.edu> wrote:
> So the major difference between what I said and what the Federal Circuit
> quoted with approval is that I talked about the "only uses" and they
> said "sole purpose".

Wrong again.

Roche put a limit on the ``philosophical experimentation'' form of
experimental use.

You claimed that this was a limit on all forms of experimental use.

> As to "to acertain the verity and exactness of the specification," a quick
> search of USPQ (from Volume 80 to present) doesn't find that quote.

Sawin v. Guild, 21 F. Cas. 554, 555 (C.C.D. Mass. 1813). There hasn't
been any dispute on the topic since then. See, e.g., Eisenberg, Patents
and the Progress of Science, 56 U. Chicago Law Review 1017, 1074 (1989).

---Dan
Let your users manage their own mailing lists. http://pobox.com/~djb/qmail.html

---

Subject: Re: "Experts slam PGP encryption"
Date: 31 Mar 1997 02:59:45 GMT
From: hollaar@ursa0.cs.utah.edu (Lee Hollaar)
Message-ID: <5hn9b1$rj2@magus.cs.utah.edu>
References: <1997Mar2503.30.53.14543@koobera.math.uic.edu>
Newsgroups:
alt.security.pgp,comp.security.pgp.discuss,alt.politics.org.nsa,talk.politics.crypto
Lines: 76

In article <1997Mar2503.30.53.14543@koobera.math.uic.edu> djb@koobera.math.uic.edu
(D. J. Bernstein) writes:
>Lee Hollaar <hollaar@ursa0.cs.utah.edu> wrote:
>> So the major difference between what I said and what the Federal Circuit
>> quoted with approval is that I talked about the "only uses" and they
>> said "sole purpose".
>
>Wrong again.
>
>_Roche_ put a limit on the ``philosophical experimentation'' form of
>experimental use.
>
>You claimed that this was a limit on all forms of experimental use.
>
>> As to "to acertain the verity and exactness of the specification," a quick
>> search of USPQ (from Volume 80 to present) doesn't find that quote.
>
>_Sawin v. Guild_, 21 F. Cas. 554, 555 (C.C.D. Mass. 1813). There hasn't
>been any dispute on the topic since then. See, e.g., Eisenberg, Patents
>and the Progress of Science, 56 U. Chicago Law Review 1017, 1074 (1989).

Thanks for the cites. I'll look them up.

But keep in mind that I wasn't writing a treatise on experimental use,
but addressing the point that Mike McCarty made --
Patents do not prevent "making". They prevent making for another's use.
One can make and use anything.
and
From my patent attorney when I got my first patent. He pretty well
spelled out what we could prevent, and what we could not. He made it
plain that we could not prevent people from making what we had and
experimenting and using it for their own purposes.

I pointed out the language of the patent statutes make making and using an
infringement, but that there was a narrow exception for experimental use
and quoted a key case on the matter (_Roche_).

And nothing that has been said in this discussion has shown that there is
a general exception to patent infringement for personal use, or even all
experimental use.

I just reread _Roche_, and there is nothing I see indicating that the court was
limiting its views on experimental use to only "philosophical experimentation"
rather than experimental use in general. But certainly making or using a
patented invention to acertain the verity and exactness of the specification"
could be regarded as "for amusement, to satisfy idle curiosity or for strictly
philosophical inquiry", especially when not done for commercial purposes.

As to experimental use for a commercial purpose, _Roche_ says:
We cannot construe the experimental use rule so broadly as to allow a
violation of the patent laws in the guise of "scientific inquiry," when
that inquiry has definite, cognizable, and not insubstantial commercial
purposes.

So there is a reasonable argument that _Roche_, decided in 1984, limits
experimental use in any form for commercial purposes, refining the 1813
decision you cited.

There is nothing in _Roche_ to indicate the court was addressing just one of
two types of experimental use defenses to infringement. There is nothing
inconsistent, in my view, between _Roche_ and experimental use to determine
whether a patented invention actually works, at least in a noncommercial
setting. Certainly one reason to "satisfy idle curiosity or for strictly
philosophical inquiry" would be seeing a patent and being surprised that the
thing even works, then trying it just to see that it does.

To the extent that other commentators may view the holding in _Roche_ as
limited to a particular type of experimental use, they are welcome to their
opinion, but their opinion doesn't make mine false or wrong.

Lee Hollaar

ps. If you really want to flog this some more, I suggest we do it by email
since this has gone well beyond my pointing out that you can infringe a
patent by personal use and probably beyond the interest of the people in
this newsgroup.

Subject: Re: "Experts slam PGP encryption"
Date: Mon, 31 Mar 1997 11:03:38 -0800
From: David Sternlight <david@sternlight.com>
Message-ID: <33400A89.7DB7@sternlight.com>
References: <5hn9b1$rj2@magus.cs.utah.edu>
Newsgroups:
alt.security.pgp,comp.security.pgp.discuss,alt.politics.org.nsa,talk.politics.crypto
Lines: 16

Lee Hollaar wrote:

<more on what patent law does and does not provide, omitted.>

> ps. If you really want to flog this some more, I suggest we do it by email
> since this has gone well beyond my pointing out that you can infringe a
> patent by personal use and probably beyond the interest of the people in
> this newsgroup.

Please don't. An intelligent, reasoned, calm discussion of some of the
details of patent law relevant to the topic here, with the intention of
gaining specific clarity and with citations and quotes, is just what
this place needs instead of the unbuttoned attempts at proof by
assertion by those repeating inaccurate "impressions".

David

Subject: Re: "Experts slam PGP encryption"
Date: Tue, 25 Mar 1997 00:58:25 -0800
From: David Sternlight <david@sternlight.com>
Message-ID: <333793B0.25AF@sternlight.com>
References: <5h79b7$c6j@magus.cs.utah.edu>
Newsgroups:
alt.security.pgp,comp.security.pgp.discuss,alt.politics.org.nsa,talk.politics.crypto
Lines: 40

Lee Hollaar wrote:
>
> In article <33357917.1FE9@sternlight.com> david@sternlight.com writes:
> Somehow, the original Bernstein article did not make it to my machine,
> so I'll go with the sternlight response ...
>
> >D. J. Bernstein wrote:
> >>
> >> Lee Hollaar <hollaar@ursa0.cs.utah.edu> wrote:
> >> > In that case, the court said that the only uses that fall within the
> >> > exception are those made "for amusement, to satisfy idle curiosity
> >> > or for strictly philosophical inquiry."
> >>
> >> False.
> >>There are two allowed forms of experimental use. The first is
> >> ``to ascertain the verity and exactness of the specification.'' The
> >> second, which _Roche_ called ``truly narrow'' and defined as you said,
> >> is ``philosophical experimentation.''
>
> Here is the actual quote from _Roche Products v. Bolar Pharmaceutical_:
> [B]y 1861, the law was "well-settled that an experiment with a
> patented article for the sole purpose of gratifying a philosophical
> taste, or curiosity, or for mere amusement is not an infringement
> of the rights of the patentee." Peppenhausen v. Falke, 19 F. Cas.
> 1048, 1049 (C.C.S.D.N.Y. 1861)
>
> So the major difference between what I said and what the Federal Circuit
> quoted with approval is that I talked about the "only uses" and they
> said "sole purpose".
>
> As to "to acertain the verity and exactness of the specification," a quick
> search of USPQ (from Volume 80 to present) doesn't find that quote. Could
> you please furnish a cite, particularly to a Federal Circuit case or one
> adopted by them?
> Lee

Thanks for the quote. I note with interest that the court said "an"
experiment, and not "experiments".

David

Subject: Re: "Experts slam PGP encryption"
Date: 29 Mar 1997 22:34:34 GMT
From: hollaar@ursa0.cs.utah.edu (Lee Hollaar)
Message-ID: <5hk5dq$st3@magus.cs.utah.edu>
References: <1997Mar2317.54.37.8375@koobera.math.uic.edu>
Newsgroups:
alt.security.pgp,comp.security.pgp.discuss,alt.politics.org.nsa,talk.politics.crypto
Lines: 33

In article <1997Mar2317.54.37.8375@koobera.math.uic.edu> djb@koobera.math.uic.edu
(D. J. Bernstein) writes:
>Lee Hollaar <hollaar@ursa0.cs.utah.edu> wrote:
>> In that case, the court said that the only uses that fall within the
>> exception are those made "for amusement, to satisfy idle curiosity
>> or for strictly philosophical inquiry."
>
>False. There are two allowed forms of experimental use. The first is
>``to ascertain the verity and exactness of the specification.'' The
>second, which _Roche_ called ``truly narrow'' and defined as you said,
>is ``philosophical experimentation.''

I couldn't find a case with the quote "verity and exactness of the
specification" when I did a search of USPQ and USPQ2d. Could you
provide a cite to a specific case?

Also, neither form for experimental use would provide any exception
for somebody making or using the invention without permission of the
patent owner for personal use. The first exception would no longer
exist when you have first determined that the invention works.

Also, in patent law there are two very distinct means for "experimental
use." The one I have been discussing is when somebody other than the
patent owner makes or uses the patented invention to determine something
about it. It's a very narrow, court-made exception to patent infringement.

The other "experimental use" is when the inventor makes the invention
available to others to determine whether it actually works. The
question is whether this starts the one-year public-use statutory bar.
In one case, the paving of a public highway was found not to be a public
use. In another case, the use of a corset by the inventor's girlfriend
was a public use.

Lee Hollaar

Terry Ritter, *his current address*, *and his top page*.

Last updated: 1999-01-19

AES and patent rights

A *Ciphers By Ritter* Page

Terry Ritter

A major discussion about the Advanced Encryption Standard and individual patent rights.

First: Crypto manufacturers have found a way to get society to pay *twice* for a new cipher: first for the research and development, and then again for the software to run it. Is this country great or what?

Then: Is a web page considered "publication"?

Contents

- 1998-09-26 Bruce Schneier: "Why should NIST consider options that cost when there are many--and some very good ones--that don't cost?" "I don't believe that it is un-American, unconstitutional, or inappropriate for automobile companies to sponsor race cars, either."
- 1998-09-26 Terry Ritter: "Really? You would force everyone who entered a car in the race to sign over their rights to their design -- including any new innovations -- if they won? That sounds like a very strange race to me."
- 1998-09-26 Bruce Schneier: "NIST is not taking anything without compensation. Everything is being given freely. You are not being compelled to submit and to give up your rights."
- 1998-09-27 Douglas A. Gwyn: "...if people can obtain free technology that provides adequate security and meets their other requirements, then of course there is not much incentive to develop competing security technology, except as an intellectual pastime."
- 1998-09-27 Terry Ritter: "A standard *cipher* should be an advantage for bankers who want the liability protection of "due diligence." "But companies and individuals can make their own decisions about what cipher to use, based on the opinions of experts they trust, or just random chance. Freedom is like that."
- 1998-09-28 Bruce Schneier: "The analogy was not very good; it wasn't worth defending. I was thinking of auto racing sponsorship as something done for publicity; you pointed out that the auto manufacturers got to keep rights to their designs. Perhaps better would be the sponsorship of a particular olympics team. When Company X sponsors the U.S. ski team, they spend money and expertise (or at least, buy expertise) and receive nothing back except publicity."
- 1998-09-28 STL137: "Har har - this one goes on my quotes page..."
- 1998-09-28 Terry Ritter: "...a government process -- one that should apply to me just the same as you -- would cost me *more* than it cost you. This is just not equal protection under the law." "My stuff is available to anyone. It is not restricted to students who are physically close to a major library system. It doesn't cost anything at all, not the price of a symposium, not the price of a book, not even the price of a CD. In my view, this is the way Science should be, and that is the way I do it."
- 1998-09-27 Jim Gillogly: "How is this not equal protection under the law? You've simply chosen a different path. Have you yet answered the question about what IBM has received for DES beyond publicity?"
- 1998-09-28 Terry Ritter: "We do know that intensive negotiation was going on right up until DES was actually published. Negotiation normally means that some form of compromise or reciprocal bargain is being struck. It simply does not take much negotiation to say: 'We'll just make this cipher free for use in the US.'"
- 1998-09-28 Bruce Schneier: "The difference seems to me that we see that participation is a choice, so there is no taking. He seems to feel differently."
- 1998-09-28 John Savard: "Obviously, reading your book _Applied Cryptography_ will lead people to suspecting that you are one of the members of this 'conspiracy' as well."
- 1998-09-28 Jim Gillogly: "I don't disagree that designing an important algorithm such as RSA is a significant piece of intellectual property and the authors deserve a reward. I do disagree that the patent process is set up to offer that reward."
- 1998-09-28 lamontg@bite.me.spammers: "It is not a contest that you care to enter, clearly. And you're clearly upset because you thought you'd get a whole lot of money out of possibly winning the contest, and you won't."
- 1998-09-29 Terry Ritter: "Had you been around here when this all started, you would know that I have never had any delusions about possibly *winning*. I did have the delusion that I would be able to *participate* without giving away my second-born child."
- 1998-09-28 John Savard: "What it is instead doing by means of the AES process is: searching for a good cipher algorithm that the U.S. government can _use in practice_ for safeguarding unclassified but sensitive communications."
- 1998-09-29 Douglas A. Gwyn: "When government officials get caught engaging in such illegal procurement activities, the penalties can be severe."
- 1998-09-29 W T Shaw: "Are you naive enough to not see what goes on. So much bidding is only done to present the guise of a credible process."
- 1998-09-28 Bruce Schneier: "To me it looks like we were both given the same decision to make, and you chose one path and I chose the other. You believed that your patent rights were worth more than NIST was willing to give you for them. "
- 1998-09-29 Terry Ritter: "This is sort of a strange comment, isn't it? It might even be the basis for a sort of occupational joke, where a mathematician gets 'paid' with zero dollars and goes away satisfied! Ha ha, very funny!"
- 1998-09-29 A [Temporary] Dog: "...why do you care if you participate or not?"
- 1998-09-29 Terry Ritter: "Well, I was disappointed. But the reason I *care* is that I think it is *wrong*." "Anyone deserves prestige for the quality of their work. But they also deserve to be compensated for *doing* the work."
- 1998-09-29 Patrick Juola: "Wrong, sir!"
- 1998-09-29 Douglas A. Gwyn: "Prestige is awarded by others, not by yourself."
- 1998-09-30 JPeschel: "Crazy man -- can you dig it! When do we do zap the capitalist system into a blue way-gone haze?"
- 1998-09-30 Lenz: "Any compensation for cipher design needs to come from winning in the marketplace, not in AES." "That means that any lack of compensation is just an indication of failure in the marketplace, which you probably would not want to claim for your designs."
- 1998-09-30 bryanolson@my-dejanews.com: "NIST correctly predicted that the worlds best cryptographers would offer top quality designs for free."
- 1998-09-30 Lincoln Yeoh: "I see AES as a crypto donation drive- 'Donate Decent Crypto to the World'."
- 1998-09-30 Stephen M. Gardner: "Many would say that when you do something just for money you cheapen it. After all, when you have the talent it just has to come out whether someone is willing to pay or not."
- 1998-10-01 Mark Wooding: "Creativity exists because, fundamentally, people enjoy being creative. That ought to be enough. It's a real shame it isn't."
- 1998-10-01 W T Shaw: "When it comes to crypto, some other areas too, things are twisted upside down from the classic, nice talk about free enterprise, so the government becomes the fixer, manipulator, and irreverant boss who ultimately selects blesses the commercial endeavors that submit to the rules."
- 1998-09-29 Douglas A. Gwyn: "I think a key (tacit) element in Terry's reasoning is that AES will be widely used, even mandatory in some cases, displacing commercial systems that might have been used instead."
- 1998-09-30 Joseph K. Nilaad: "I don't think we're all working for charity!"
- 1998-09-30 R H Braddam: "...just because the encryption algorithm is free don't mean that the applications using them will be."
- 1998-09-30 Bruce Schneier: "Presumably some of the AES submissions have patent applications pending."
- 1998-10-01 dianelos@tecapro.com: "FROG is in the public domain too."
- 1998-09-30 Patrick Juola: "The Olympics certainly pay 'only the medal,' yet have no problem getting competitors."

- 1998-10-03 Lincoln Yeoh: "...most people don't get paid in pats on backs, praises, respect etc, but it actually matters a lot to many people."
- 1998-09-30 Joseph K. Nilaad: "Thanks for bringing Olympic scenario up, who is collecting the money? Certainly not the atheletes. I have not yet seen any free admission in Olympic."
- 1998-10-1 R H Braddam: "...just because the other submissions were not YET patented, that doesn't mean they were unpatentable, or worthless."
- 1998-10-2 Joseph K. Nilaad: "Free worldwide and 40bit export law at this moment and in this country, is a contradiction!"
- 1998-10-2 R H Braddam: "The original ITAR export controls were very likely unconstitutional, and the movement of control to Department of Commerce was just a stop-gap measure to delay or circumvent having all or a majority of them thrown out by the Supreme Court."
- 1998-10-02 Casper H.S. Dik - Network Security Engineer: "While this has been made to sound admirable, it really was only a ploy to keep the 'underclass' who couldn't possibly afford to compete for free away. Women weren't allowed to compete either."
- 1998-09-30 lamontg@bite.me.spammers: "if i go out and do something which i consider to be work and everyone else considers to be worthless then i certainly don't deserve to be compensated for doing the work."
- 1998-09-29 Bruce Schneier: "I don't think I have a "point," other than you chose not to submit and I chose to submit."
- 1998-09-29 Andreas Enterrottacher: "You didn't make money with DES and you won't make money with the next standard. NIST didn't need you to get DES and they don't need you to get AES :)"
- 1998-09-29 Terry Ritter: "...while there may be some "free" programs which use AES, we can be sure that commercial software firms will compensate their programmers by charging for the software. Programmers thus will be compensated -- and justly so -- for the time they spend; but cipher designers will *not* be compensated for the vastly greater time *they* spend."
- 1998-09-30 Andreas Enterrottacher: "...since DH has become free RSA isn't used any more in PGP. As well others don't use RSA because DH has become free."
- 1998-09-30 John Savard: "...RC4 is the *beneficiary* of a government monopoly..."
- 1998-09-28 Bruce Schneier: "...it takes a lot of work to keep with the Internet, and some might argue that it is impossible. There is just so much out there, and such a high percentage of it is trash, that it just isn't cost effective to wade through it all. For everyone like you, who is doing serious work, there are dozens of yahoos who aren't. And from first glance, you can't tell the difference."
- 1998-09-28 Lincoln Yeoh: "Don't go to the Crypto Donation Drive, if you don't want to give."
- 1998-09-28 Lamont Granquist: "*You* are the one that is lazy. Writing an article and sticking it up on a website is something that any college student can do. Doing the work to make it suitable for publication in a journal takes substantially longer. That standard of quality is why 'lazy' cryptographers read journals rather than browse random web pages."
- 1998-09-28 W T Shaw: "It all depends on whether you actually want to keep up with a fast changing field or not; this affects everything, not cryptography alone. I would suggest that you discard email as well since, according to the same reasoning, nothing not carefully written out and properly mailed would constitute laziness."
- 1998-09-30 Lamont Granquist: "Saying 'I put it up on my web site 6 months ago, why haven't you read it?' is egotistical b.s. and it isn't publishing -- getting it into central repositories of information is publishing."
- 1998-09-28 W T Shaw: "We are at a great transition where actual printed matter is fastly being usurped by electronic media...it's in all the *papers*. Holding to the old standard as the one true path is merely quaint."
- 1998-09-28 Bruce Schneier: "Anyone can put their ideas up on the web; it's the ultimate vanity press. But there is just too much out there; the community needs some way to determine if a particular something is worth reading."
- 1998-09-29 Douglas A. Gwyn: "If it makes a good example, it is pedagogically useful to have it in your article, regardless of its origin."
- 1998-09-29 Bruce Schneier: "...this is the reality of academic cryptography. I cannot change that, even if I sit on program committees and argue the point."
- 1998-09-29 Joseph K. Nilaad: "How can you quantify which publication is worth reading either it is on the web or hard copy?"
- 1998-09-29 Patrick Juola: "...NIST has something that most people don't -- to wit, credibility."
- 1998-09-29 Bruce Schneier: "If academic cryptographer A reads and cryptanalyzes a method that appears at a conference, workshop, or as an AES submission, he can publish his results. If he reads and cryptanalyzes a method that appears on Usenet, all he can do is post his results on Usenet."
- 1998-09-30 Terry Ritter: "...now that I think of it, Biham *actually* *did* write and publish an academic "paper" on *my* *own* "Ladder DES" proposal, which was basically a Usenet thing." "So, clearly, one *can* write about Usenet proposals."
- 1998-09-30 Bruce Schneier: "Of course one can. Biham's paper is certainly a good example."
- 1998-09-29 Douglas A. Gwyn: "The issue isn't whether or not you can access the document on-line, it's whether or not the document has made it through a reasonable 'antijunk' filter."
- 1998-09-30 Stephen M. Gardner: "when something is not broken yet and it hasn't had a lot of attempts made yet it is really not secure."
- 1998-09-28 Paul Rubin: "What I remember is hearing that it's hard for a cipher designer to be taken seriously until they've gotten some serious, interesting results at breaking other people's algorithms."
- 1998-09-28 W T Shaw: "The official line was that you had bettter not even try; if you wanted to think in the area, you had better register; and don't think of implementing anything without going through some sort of Greek Debate on its utility first."
- 1998-09-29 Douglas A. Gwyn: "There undoubtedly is some good unreviewed Web publication, as well as a lot of bogosity. And there are also dubious or even bogus peer-reviewed articles."
- 1998-09-29 David A. Scott: "...very little in crypto in this country and else where is not poisioned by the self serving NSA."
- 1998-09-29 W T Shaw: "...I see so many in NIST going out of their way to be fair even if it hurts for as long as their reins will let them run that way."
- 1998-09-30 John Savard: "Well, the NSA has a proud history which includes sinking a lot of Nazis...."
- 1998-09-28 RREYNARD: "...as a retired businessman, it appears that the creator/inventor of cryptographic algorithms and systems finds himself in the unfortunate position of being in the wrong business at the wrong time."
- 1998-09-28 Jerry Leichter: "...there's a general legal principle, called I think laches, which basically says that if you arbitrarily delay raising a legal claim for too long, you may lose your right to do so."
- 1998-09-29 Terry Ritter: "A patent is an offensive right, yes, but it is a right to collect damages." "Indeed, until a particular cipher is picked, there is likely to be little or no damage at all."
- 1998-09-29 John Savard: "Plus, if waiting for someone with deep pockets to sue constitutes an invalidation of patent rights, then so would waiting for someone to sue with small pockets for hiring a lawyer..." "*Several* of the AES candidates may be infringing the IDEA patent..."
- 1998-09-29 Bruce Schneier: "I will look at the IDEA patent."
- 1998-09-29 W T Shaw: "The government reserves the right to refuse to approve a standard based on the entries and adopt something else, or not set any standard...and not tell why."
- 1998-09-27 Lamont Granquist: "If *ANYTHING* is unconstitutional and un-american it is giving the winner of this contest a guaranteed 15 or so year contract to reap royalties from every government-used encryption product out there. It's like saying that all the cars the government uses for the next 15 years will be Fords."
- 1998-09-27 Terry Ritter: "Personally, I would have been willing to give *the government* a very attractive license. But that is not enough for our government: AES demanded that I give a free license to every for-profit company who would be *selling* software based on that technology. That sure sounds like a government-mandated subsidy for those guys, doesn't it?" "MULTIPLE CHOICE: If Ford is the winner in a contest for the most-efficient car design, based on long-term, very expensive and privately-funded research, would we expect: a) they should be made to give their superior technology to all their competitors, free of charge, or b) they should reap just rewards for their successful research."
- 1998-09-27 Lamont Granquist: "In this case "just rewards" are the PR and cipher design expertise which is entirely sufficient for counterpane and apparently sufficient for the 15 other applicants."
- 1998-09-28 Douglas A. Gwyn: "The correct answer is 'None of the above.'"
- 1998-09-27 Bruce Schneier: "Fundamentally new techniques for the sake of themselves have no place in a conservative standard. Fundamentally new techniques that resist attacks that break other things are much more interesting."
- 1998-09-27 David A. Scott: "...I don't think the US government wants to really compensate anybodys who is not political correct...."
- 1998-09-27 John Savard: "Initially the AES committee was expressing a strong preference for royalty-free submissions; this changed to a mandatory requirement shortly after a statement by you that the preference was unlikely to create problems."
- 1998-09-27 Bruce Schneier: "Thanks for the explanation."
- 1998-09-27 David A. Scott: "I really wonder which if not all of the methods are NSA fronts."
- 1998-09-27 Bruce Schneier: "...I know that Twofish is not an NSA entry, but there is no way I can prove that to you."
- 1998-09-27 Lamont Granquist: "Yeah, twofish was designed by the NSA, fronted by Bruce and Counterpane."
- 1998-09-28 David A. Scott: "Terry I agree with you here people should be FREE to pick what they want. The AES competation is just another clipper chip in sheeps clothing. It would be best to

let market place decide on its own."

- 1998-09-28 David A. Scott: "It will be very hard to get good crypto to the masses when so called experts poo poo the idea of long keys as snake oil."
- 1998-09-28 David A. Scott: "I know you are jesting but TWOFISH would have my bet as NSA plant I just am not sure which of the rest are."
- 1998-09-28 W T Shaw: "If I were looking for plants, and I'm not, I would rather look at those who already have their hands in each others pockets."
- 1998-09-28 David A. Scott: "Terry if you did this kind of crap that Bruce did here people would not like you. But Bruce has the money and power to laugh in your face even if he knows that some of your ideas better than his."
- 1998-09-29 David A. Scott: "Of course the NSA horse will win. I think Bruce is on that horse but I am sure the NSA has more than one."
- 1998-09-29 RREYNARD: "As I recall, COBOL was the government's 'standardized' Common Business Oriented Language to be used by all government agencies for application programming. I don't believe it ever achieved 'standardization', it never became 'common' and many government agencies opted not to use it. It is my opinion, that the 'standard' crypto algorithm will realize similar success."
- 1998-09-29 David A. Scott: "I think most management mistakenly thinks that you have to pay lots of money to get something good."
- 1998-09-30 Terry Ritter: "The AES competition is government funded; presumably GNU C is not."
- 1998-09-29 W T Shaw: "Current events beat the soaps."

```
Subject: AES and patent rights
Date: Sat, 26 Sep 1998 15:41:03 GMT
From: schneier@counterpane.com (Bruce Schneier)
Message-ID: &LT;360d0782.2157160@news.visi.com>
References: &LT;360a964c.9610170@news.io.com>
   &LT;jgfunj-2409981205490001@dialup60.itexas.net>
Newsgroups: sci.crypt
Lines: 82

On Thu, 24 Sep 1998 18:58:28 GMT, ritter@io.com (Terry Ritter) wrote:
>I continue to believe that the AES requirement for submitters to give
>up their property rights by their submission was un-American,
>unconstitutional, and also inappropriate at the first stage of a
>technical comparison process.

I would tend to agree with you more if NIST received no submissions,
and nothing but protests.  But fifteen groups, including the most
aggressive crypto patent company (RSADSI), submitted algorithms.
We're all happy to give away our patent rights if we are chosen as the
standard.

Why should NIST consider options that cost when there are many--and
some very good ones--that don't cost?  That just seems smart business
sense to me.

In everything I've read about the Dept of Justice's case against
Microsoft, they never cleam that it is unconstitutional (or even
un-American) for them to give away their browser.  Their case hinges
on whether they used their Windows monopoly to manipulate the broswer
market.  Unless you can show a similar environment, I don't think you
have much of a case.

However, if you can break or otherwise show that the fifteen free
candidates are not good AES choices, NIST will have no option but to
look at patented ideas.

>I have serious alternate ciphering
>approaches, but I also *own* those approaches, and I am not going to
>simply make them "free, worldwide" until people "worldwide" start
>funding the research that brought these things forth.  If the
>government wants *my* stuff to be "free," they can first pay my
>salary, equipment, and an investment profit for a decade of work.

Indeed.  This is your choice, and you are free to make it.

>The result of this AES requirement is that those who have *not*
>invested significantly in such research (or who can afford to give it
>away), have been granted an advantage by the government over those who
>*have* invested in such research.  This is the wrong message to send.

I disagree.  I believe that RSADSI, IBM, and NTT have invested
significant resources in their submissions.  (I choose these three
because their submissions reflect those resources.)  I personally have
invested over 1000 hours of Counterpane Systems time, time that we did
not spend doing billable work, on Twofish.

We all believe two things.  One, that the collateral knowledge gained
by donating those resources is worth someting.  And two, that the PR
benefit of being chosen as AES is worth something.  I don't believe
that it was un-American, unconstitutional, or inappropriate for AT&T
to give away their rights to the transistor, or to do research on
background radiation in the universe.  I don't believe that it is
un-American, unconstitutional, or inappropriate for automobile
companies to sponsor race cars, either.

>Note that this isn't about making things free for users:  It is not
>like NIST will demand that *software* which uses AES will be "free
>worldwide."  NIST doesn't want *software* companies to lose any money
>on *their* productive efforts.  But NIST (or NSA) apparently *does*
>want cipher designers to lose money on *their* efforts.  This is
>wrong, and may be about as close to a government conspiracy to
>restrict commerce as one might want to see.

It's not NIST.  The cipher designers agreed to the rules.  Again, if
no one submitted a free candidate, then you would have a case.  There
are fifteen submitters who don't feel that the "problem" of losing
```

money for their efforts as a significant one.

>It is unfortunate that Bruce Schneier was a prime factor in getting
>the original rules changed so that only free designs would even be
>*considered* for AES.

Was I? Wow. I thought that was NIST's idea. Whatever, it seems like
the idea was a good one. As I said before, we have fifteen
submissions, some of them very good.

Bruce
****
Bruce Schneier, President, Counterpane Systems Phone: 612-823-1098
101 E Minnehaha Parkway, Minneapolis, MN 55419 Fax: 612-823-1590
Free crypto newsletter. See: http://www.counterpane.com

Subject: Re: AES and patent rights
Date: Sat, 26 Sep 1998 18:22:16 GMT
From: ritter@io.com (Terry Ritter)
Message-ID: <360d30d2.9039808@news.io.com>
References: <360d0782.2157160@news.visi.com>
Newsgroups: sci.crypt
Lines: 212

On Sat, 26 Sep 1998 15:41:03 GMT, in <360d0782.2157160@news.visi.com>,
in sci.crypt schneier@counterpane.com (Bruce Schneier) wrote:

>On Thu, 24 Sep 1998 18:58:28 GMT, ritter@io.com (Terry Ritter) wrote:
>>I continue to believe that the AES requirement for submitters to give
>>up their property rights by their submission was un-American,
>>unconstitutional, and also inappropriate at the first stage of a
>>technical comparison process.
>
>I would tend to agree with you more if NIST received no submissions,
>and nothing but protests. But fifteen groups, including the most
>aggressive crypto patent company (RSADSI), submitted algorithms.
>We're all happy to give away our patent rights if we are chosen as the
>standard.

But while you *can* give *your* *own* patent rights away, you *cannot*
give away those rights which someone else may have established. So if
someone else has established rights to what you have done, the cipher
will be just as encumbered as it might have been from you.

This means that the rule *cannot* make a cipher "free worldwide."
What it does is prohibit competition from those who have actually
invested in research and developed new technology, and who cannot
effectively benefit in other ways.

DES, for example, was free for use, but not worldwide. And even those
rights were intensely negotiated right up to the date of publication
of the standard. Note that there were many ways for the leading
computer company at the time to benefit without direct payment, from
guaranteed government contracts to reduced export hassles. We do not
know what was negotiated or hinted with a wink and a nod. But we do
know that negotiations were long and arduous. At the time, IBM took
those rights very seriously.

>Why should NIST consider options that cost when there are many--and
>some very good ones--that don't cost? That just seems smart business
>sense to me.

First, NIST is not a business. It is an arm of *my* government as
well as it is yours. *Somehow* it just happens to respect your rights
more than it does mine. *Somehow* it just happens that the rules --
that you publicly suggested and promoted before they were adopted --
favor you. Odd.

>In everything I've read about the Dept of Justice's case against
>Microsoft, they never cleam that it is unconstitutional (or even
>un-American) for them to give away their browser. Their case hinges
>on whether they used their Windows monopoly to manipulate the broswer
>market. Unless you can show a similar environment, I don't think you
>have much of a case.

While I am not overly eager to spend years of my life rectifying yet
another legal disaster, this is exactly the sort of goading that might
push me to do just that.

My argument would more likely be based on "equal protection under the
law" than antitrust; when government participation is limited by what
one owns, there have to be questions. There is also "taking without
compensation."

But I have often wondered why Microsoft does not open a new front with
Justice by supporting a challenge to AES. This would put them on the
side of the little guy, thus giving them some good press, but more
importantly could establish a precedent which could be vitally
important to their business: If AES goes through as planned, the next
government thing might be the definition of a "standard" operating

system, and only those willing to give up their rights need apply. A US standard operating system available "free worldwide" could bring down Microsoft far faster than any decades-long antitrust action.

>However, if you can break or otherwise show that the fifteen free
>candidates are not good AES choices, NIST will have no option but to
>look at patented ideas.

Once again, there is really no way to know what is patented in these submissions. All we know is that the *submitters* will give up *their* rights. But that does not mean that someone else does not have rights in those very same submissions.

If *I* were in the position where somebody had proposed a system using my owned technology, I would shut up and wait and hope to be chosen.

>>I have serious alternate ciphering
>>approaches, but I also *own* those approaches, and I am not going to
>>simply make them "free, worldwide" until people "worldwide" start
>>funding the research that brought these things forth. If the
>>government wants *my* stuff to be "free," they can first pay my
>>salary, equipment, and an investment profit for a decade of work.
>
>Indeed. This is your choice, and you are free to make it.

Thank you so much.

>>The result of this AES requirement is that those who have *not*
>>invested significantly in such research (or who can afford to give it
>>away), have been granted an advantage by the government over those who
>>*have* invested in such research. This is the wrong message to send.
>
>I disagree. I believe that RSADSI, IBM, and NTT have invested
>significant resources in their submissions. (I choose these three
>because their submissions reflect those resources.) I personally have
>invested over 1000 hours of Counterpane Systems time, time that we did
>not spend doing billable work, on Twofish.

Frankly, 1000 hours is really small change compared to the long-term development of new ciphering technology. You do use Feistel ciphering, of course, as does most everybody else. But you did not develop it.

There are various ways large companies can benefit, both from the competition and possible selection. The fact that they lose something on the design process does not mean that they will not make it up. But I suspect even they were not overjoyed at the idea of simply giving away their technology because it was good. Maybe that is *why* we see so many submissions based on old Feistel technology.

>We all believe two things. One, that the collateral knowledge gained
>by donating those resources is worth someting. And two, that the PR
>benefit of being chosen as AES is worth something. I don't believe
>that it was un-American, unconstitutional, or inappropriate for AT&T
>to give away their rights to the transistor,

This seems to be an odd comparison, since Bell Labs *licensed* those rights to Sony, which is why we had a period in the 50's and 60's when everybody had a Japanese portable transistor AM radio. And Bell Labs was not a government department.

>or to do research on
>background radiation in the universe. I don't believe that it is
>un-American, unconstitutional, or inappropriate for automobile
>companies to sponsor race cars, either.

Really? You would force everyone who entered a car in the race to sign over their rights to their design -- including any new innovations -- if they won?

That sounds like a very strange race to me.

Race drivers and their organizations have to make real money, and they depend upon the innovations in their cars. I doubt they would give up their rights -- unless of course they simply *have* no rights, and so take the opportunity to exclude their competition.

Somebody might even have the balls to take something like that to court. Especially if the race was government-sponsored.

>>Note that this isn't about making things free for users: It is not
>>like NIST will demand that *software* which uses AES will be "free
>>worldwide." NIST doesn't want *software* companies to lose any money
>>on *their* productive efforts. But NIST (or NSA) apparently *does*
>>want cipher designers to lose money on *their* efforts. This is
>>wrong, and may be about as close to a government conspiracy to
>>restrict commerce as one might want to see.
>
>It's not NIST. The cipher designers agreed to the rules. Again, if

>no one submitted a free candidate, then you would have a case.

I have a case anyway.

>There
>are fifteen submitters who don't feel that the "problem" of losing
>money for their efforts as a significant one.

There are various other reasons for someone to participate; the fact that someone participates does not imply that ownership is not a significant issue.

>>It is unfortunate that Bruce Schneier was a prime factor in getting
>>the original rules changed so that only free designs would even be
>>*considered* for AES.
>
>Was I? Wow. I thought that was NIST's idea.

Oh, please. Are we to believe you have forgotten your letter to NIST after the first conference? Shall we re-post it?

As I see it, and since you had no new technology of your own to enter, it was in your business interest to prevent everyone who had such technology from competing with you. Good business, presumably, but poor competition, and very bad science.

>Whatever, it seems like
>the idea was a good one. As I said before, we have fifteen
>submissions, some of them very good.

The competition is being conducted in a way which I believe is unconstitutional, which means that the result -- whatever it is -- will be open to challenge.

More than that, these rules act to restrict the long term development of crypto technology by not allowing fundamentally-new technology to compete, and by not rewarding the crypto design process itself. These rules are tools to minimize the open development of cryptographic technology, and every entrant who participates is another government argument that this is a good thing.

Terry Ritter ritter@io.com http://www.io.com/~ritter/
Crypto Glossary 1998-08-27: http://www.io.com/~ritter/GLOSSARY.HTM

Subject: Re: AES and patent rights
Date: Sat, 26 Sep 1998 21:37:43 GMT
From: schneier@counterpane.com (Bruce Schneier)
Message-ID: <360d5983.3024744@news.visi.com>
References: <360d30d2.9039808@news.io.com>
Newsgroups: sci.crypt
Lines: 228

I generally hate Usenet arguments. I will respond to this, and you are welcome to as many last words as you like. If there is something very new, interesting, or comment-worthy, I will respond. But I see no reason to continue volleying back and forth.

Bruce

On Sat, 26 Sep 1998 18:22:16 GMT, ritter@io.com (Terry Ritter) wrote:
>On Sat, 26 Sep 1998 15:41:03 GMT, in <360d0782.2157160@news.visi.com>,
>in sci.crypt schneier@counterpane.com (Bruce Schneier) wrote:
>
>>On Thu, 24 Sep 1998 18:58:28 GMT, ritter@io.com (Terry Ritter) wrote:
>>>I continue to believe that the AES requirement for submitters to give
>>>up their property rights by their submission was un-American,
>>>unconstitutional, and also inappropriate at the first stage of a
>>>technical comparison process.
>>
>>I would tend to agree with you more if NIST received no submissions,
>>and nothing but protests. But fifteen groups, including the most
>>aggressive crypto patent company (RSADSI), submitted algorithms.
>>We're all happy to give away our patent rights if we are chosen as the
>>standard.
>
>But while you *can* give *your* *own* patent rights away, you *cannot*
>give away those rights which someone else may have established. So if
>someone else has established rights to what you have done, the cipher
>will be just as encumbered as it might have been from you.
>
>This means that the rule *cannot* make a cipher "free worldwide."
>What it does is prohibit competition from those who have actually
>invested in research and developed new technology, and who cannot
>effectively benefit in other ways.

Indeed. This is true, and worrisome. It is possible that one (or more) of the AES submissions infringes on some patent (or some pending patent) held by some third party, and that third party will not say anything until it is too late. This has happened in other computer

standards committees, and the results are generally disasterous. All we can do is to hope for the best.

But yes, I do think about this and I am concerned.

>DES, for example, was free for use, but not worldwide. And even those
>rights were intensely negotiated right up to the date of publication
>of the standard. Note that there were many ways for the leading
>computer company at the time to benefit without direct payment, from
>guaranteed government contracts to reduced export hassles. We do not
>know what was negotiated or hinted with a wink and a nod. But we do
>know that negotiations were long and arduous. At the time, IBM took
>those rights very seriously.

From what I have been researched, IBM has never sued or even threatened anyone for using DES. If you have other evidence, I very much want to hear it?

>>Why should NIST consider options that cost when there are many--and
>>some very good ones--that don't cost? That just seems smart business
>>sense to me.
>
>First, NIST is not a business. It is an arm of *my* government as
>well as it is yours. *Somehow* it just happens to respect your rights
>more than it does mine. *Somehow* it just happens that the rules --
>that you publicly suggested and promoted before they were adopted --
>favor you. Odd.

Probably a government conspiracy, that's what I think.

Although more likely the government didn't want to force users of AES to pay royalties, when there was the very strong possibility that free alternatives migh be out there. So NIST took a risk in only asking for unencumbered submissions, but it looks like their risk paid off. You and I and everyone else who builds encryption systems using AES will benefit.

>My argument would more likely be based on "equal protection under the
>law" than antitrust; when government participation is limited by what
>one owns, there have to be questions. There is also "taking without
>compensation."

NIST is not taking anything without compensation. Everything is being given freely. You are not being compelled to submit and to give up your rights.

>>However, if you can break or otherwise show that the fifteen free
>>candidates are not good AES choices, NIST will have no option but to
>>look at patented ideas.
>
>Once again, there is really no way to know what is patented in these
>submissions. All we know is that the *submitters* will give up
>*their* rights. But that does not mean that someone else does not
>have rights in those very same submissions.
>
>If *I* were in the position where somebody had proposed a system using
>my owned technology, I would shut up and wait and hope to be chosen.

I know. You and many others. All we can do is hope. There is some caselaw on the subject. NIST will make a public call to all third parties to state any potential patent claims regarding the submissions. If someone chooses not to, NIST could argue in court that the patentholder deliberately withheld information in an attempt to hide his rights until after AES was awarded. Will this do any good? No one knows.

>>>The result of this AES requirement is that those who have *not*
>>>invested significantly in such research (or who can afford to give it
>>>away), have been granted an advantage by the government over those who
>>>*have* invested in such research. This is the wrong message to send.
>>
>>I disagree. I believe that RSADSI, IBM, and NTT have invested
>>significant resources in their submissions. (I choose these three
>>because their submissions reflect those resources.) I personally have
>>invested over 1000 hours of Counterpane Systems time, time that we did
>>not spend doing billable work, on Twofish.
>
>Frankly, 1000 hours is really small change compared to the long-term
>development of new ciphering technology. You do use Feistel
>ciphering, of course, as does most everybody else. But you did not
>develop it.

Of course, every cryptography algorithm builds on the work of others. I was only talking about direct Twofish development. We used Feistel networks (invented by Feistel), key-dependent S-boxes (I first saw this in Khufu by Raph Merkle), S-boxes build out of a combination of fixed S-boxes and a linear operation (basically, a rotor machine), MDS matrices (used in Square, researched by Serge Vaudenay, and etc), pseudo-Hadamard transforms (invented by Jim Massey), the idea of mixing operations from different groups (researched by Xujia Lai, and then by lots of other people), and analytic techniques for differential, linear, higher-order differential, interpolation, related-key, and other cryptanalyses (invented and extended by Eli Biham, Adi Shamir, Lars Knudsen, Vincent Rijmen, us, Carlo Harpes, Thomas Jakkobson, Mitsuru Matsui, Shiho Morai, and countless others).

Let it never be said that we did our work in a vacuum.

>There are various ways large companies can benefit, both from the
>competition and possible selection. The fact that they lose something
>on the design process does not mean that they will not make it up.
>But I suspect even they were not overjoyed at the idea of simply
>giving away their technology because it was good.

I suspect you are wrong, but that may be a false impression that I got
from having conversations on this topic with them.

>Maybe that is *why*
>we see so many submissions based on old Feistel technology.

No, the reason is because it is a good technology. Decoupling the
design of the F-function from the encryption/decryption strucuture is
nice, as is not having to worry about things working in the reverse
direction. What is interesting to me is that we saw some of the never
variations of the Feistel network: target-heavy unbalanced Feistel
networks in MARS, and incomplete Feistel networks in CAST-256. Even
the E2 work shows that there are still things to learn about the
Feistel structure.

>>There
>>are fifteen submitters who don't feel that the "problem" of losing
>>money for their efforts as a significant one.
>
>There are various other reasons for someone to participate; the fact
>that someone participates does not imply that ownership is not a
>significant issue.

Again, you may be right. All I have done is talk with the people
involved.

>>>It is unfortunate that Bruce Schneier was a prime factor in getting
>>>the original rules changed so that only free designs would even be
>>>*considered* for AES.
>>
>>Was I? Wow. I thought that was NIST's idea.
>
>Oh, please. Are we to believe you have forgotten your letter to NIST
>after the first conference? Shall we re-post it?

I believe you are confusing my endorsement of an idea with my
origination of an idea. I find it hard to believe that I imposed my
desire for a free standard onto a reluctant NIST. If so, good for me.
But I remember NIST wanting to ensure that AES was unencumbered by
patents. Of course I agree with that position, and of course I said
so in my letter to NIST. Thanks for the compliment, all the same.

>As I see it, and since you had no new technology of your own to enter,
>it was in your business interest to prevent everyone who had such
>technology from competing with you. Good business, presumably, but
>poor competition, and very bad science.

On the contrary, I am seeing some excellent science. But as I said
before, you are welcome to break the submissions and show all of us
wrong. In all honesty, if you can do that people will be willing to
pay for the patented technology that your techniques cannot break.

(I'll also pay for a strong bock cipher that encrypt data (ECB mode)
in less than 2 clock cycles per byte. I don't know how to do that.)

>>Whatever, it seems like
>>the idea was a good one. As I said before, we have fifteen
>>submissions, some of them very good.
>
>The competition is being conducted in a way which I believe is
>unconstitutional, which means that the result -- whatever it is --
>will be open to challenge.

Unconstitutional!? Neat. Which part of the constitution do you see
being trampled on in this competition?

>More than that, these rules act to restrict the long term development
>of crypto technology by not allowing fundamentally-new technology to
>compete, and by not rewarding the crypto design process itself. These
>rules are tools to minimize the open development of cryptographic
>technology, and every entrant who participates is another government
>argument that this is a good thing.

As a crypto designer, I have found the whole process more rewarding
than almost anything else I have done. I expect those rewards to
continue, even if Twofish does not get chosen.

And honestly, in a standard I would rather see a conservative design
than fundamentally new technology. If you have the latter, start
writing papers and getting your ideas into the literature. Time needs
to test things; this is cryptography, after all.

Bruce
****
Bruce Schneier, President, Counterpane Systems Phone: 612-823-1098
101 E Minnehaha Parkway, Minneapolis, MN 55419 Fax: 612-823-1590

Free crypto newsletter. See: http://www.counterpane.com

Subject: Re: AES and patent rights
Date: Sun, 27 Sep 1998 07:45:15 GMT
From: "Douglas A. Gwyn" <DAGwyn@null.net>
Message-ID: <360DECE4.33B6870C@null.net>
References: <360d5983.3024744@news.visi.com>
Newsgroups: sci.crypt
Lines: 32

Bruce Schneier wrote:
> Probably a government conspiracy, that's what I think.

Absolutely! Everything anybody doesn't like must be due
to some evil government consipracy. Could there possibly
be any other explanation?

> On the contrary, I am seeing some excellent science. But as I said
> before, you are welcome to break the submissions and show all of us
> wrong. In all honesty, if you can do that people will be willing to
> pay for the patented technology that your techniques cannot break.
> (I'll also pay for a strong bock cipher that encrypt data (ECB mode)
> in less than 2 clock cycles per byte. I don't know how to do that.)

The way I would put this is, if people can obtain free technology
that provides adequate security and meets their other requirements,
then of course there is not much incentive to develop competing
security technology, except as an intellectual pastime.
But that's not a conspiracy, it's simple economics.

> Unconstitutional!? Neat. Which part of the constitution do you see
> being trampled on in this competition?

You're right; the US constitution (unfortunately) does not
prohibit government intervention in the marketplace. That's
how we migrated from an essentially laisse-faire capitalist
society into a mixed economy.

If freedom of speech, or the right to bear arms, or due
process, etc. were involved, then there would be a
constitutional issue, but so far as I can see there is
no significant problem along those lines involving AES.

Subject: Re: AES and patent rights
Date: Sun, 27 Sep 1998 22:34:37 GMT
From: ritter@io.com (Terry Ritter)
Message-ID: <360ebd79.18691713@news.io.com>
References: <360d5983.3024744@news.visi.com>
Newsgroups: sci.crypt
Lines: 232

On Sat, 26 Sep 1998 21:37:43 GMT, in <360d5983.3024744@news.visi.com>,
in sci.crypt schneier@counterpane.com (Bruce Schneier) seemed to
respond, yet failed to address his own analogy:

>[...]
>I don't believe that it is
>>un-American, unconstitutional, or inappropriate for automobile
>>companies to sponsor race cars, either.
>
>Really? You would force everyone who entered a car in the race to
>sign over their rights to their design -- including any new
>innovations -- if they won?
>
>That sounds like a very strange race to me.
>
>Race drivers and their organizations have to make real money, and they
>depend upon the innovations in their cars. I doubt they would give up
>their rights -- unless of course they simply *have* no rights, and so
>take the opportunity to exclude their competition.
>
>Somebody might even have the balls to take something like that to
>court. Especially if the race was government-sponsored.

>[...]
>From what I have been researched, IBM has never sued or even
>threatened anyone for using DES. If you have other evidence, I very
>much want to hear it?

Please try to follow along: DES was a US standard. It was free for
use in the US. Presumably IBM got something for that. Lawsuits and
threatening have nothing to do with it.

>[...]
>Although more likely the government didn't want to force users of AES
>to pay royalties, when there was the very strong possibility that free
>alternatives migh be out there. So NIST took a risk in only asking
>for unencumbered submissions, but it looks like their risk paid off.
>You and I and everyone else who builds encryption systems using AES

>will benefit.

A standard *cipher* should be an advantage for bankers who want the liability protection of "due diligence."

But companies and individuals can make their own decisions about what cipher to use, based on the opinions of experts they trust, or just random chance. Freedom is like that.

On the other hand, a government *interface* standard which could handle (virtually) any cipher of any sort as dynamically selected, would be useful.

>>My argument would more likely be based on "equal protection under the
>>law" than antitrust; when government participation is limited by what
>>one owns, there have to be questions. There is also "taking without
>>compensation."
>
>NIST is not taking anything without compensation. Everything is being
>given freely. You are not being compelled to submit and to give up
>your rights.

Indeed, I did not submit.

But *you* get to participate in a government-funded process which took nothing from you, but would take property from me.

This is a little more than "not being compelled to submit."

>>[...]
>>If *I* were in the position where somebody had proposed a system using
>>my owned technology, I would shut up and wait and hope to be chosen.
>
>I know. You and many others.

Life in the property lane: You don't own the land you travel over unless you got there first and registered your claim. Deal with it.

>All we can do is hope.

Exactly how *I* feel! The whole process could be *quite* amusing.

It's something like a small herd of nearsighted bumbling sheep with their own blind shepherd.

>There is some
>caselaw on the subject. NIST will make a public call to all third
>parties to state any potential patent claims regarding the
>submissions. If someone chooses not to, NIST could argue in court
>that the patentholder deliberately withheld information in an attempt
>to hide his rights until after AES was awarded. Will this do any
>good? No one knows.

As far as I know, there is *no* responsibility in patents to take offensive action at any particular time *or* to respond to governmental calls for clarification. Perhaps you are thinking of copyright.

>[...]
>>>>It is unfortunate that Bruce Schneier was a prime factor in getting
>>>>the original rules changed so that only free designs would even be
>>>>*considered* for AES.
>>>
>>>Was I? Wow. I thought that was NIST's idea.
>>
>>Oh, please. Are we to believe you have forgotten your letter to NIST
>>after the first conference? Shall we re-post it?
>
>I believe you are confusing my endorsement of an idea with my
>origination of an idea.

I'm surprised you aren't crying "I don't remember, I can't recall." Of course that would do no good, since all this is in the public record, but at least it would be more amusing than you blaming me for your words.

The NIST position coming out of the first conference was that "patented algorithms" (sic) would be accepted, but there would be a bias against them. You then argued in a letter -- which you made public -- that patented algorithms should not be accepted at all. And that became the rule. And of course it was just a coincidence that this also stripped off some of your competition.

Are you in charge of NIST? No. Do you bear ultimate responsibility? No. Would you be the target of a lawsuit? Probably not. Was your letter a major factor in the internal debate? My guess is yes. So, are you the proximate cause of the rules change? Probably so.

>I find it hard to believe that I imposed my

>desire for a free standard onto a reluctant NIST.

Everybody has limited resources, so we all want everything free. Many of us are marketplace infants or defeated warriors who would prefer to suck at the government teat. Unfortunately, there are social consequences to free things, and those consequences lead to less funding for cipher design and technology. I see this as a bad trade for society. In the end, it is better to actually pay the worth of production, rather than to "depend upon the kindness of strangers."

>If so, good for me.
>But I remember NIST wanting to ensure that AES was unencumbered by
>patents. Of course I agree with that position, and of course I said
>so in my letter to NIST. Thanks for the compliment, all the same.

>[...]
>(I'll also pay for a strong bock cipher that encrypt data (ECB mode)
>in less than 2 clock cycles per byte. I don't know how to do that.)

And once you know, would you still pay?

Certainly my stuff can do 1 cycle PER BLOCK with enough hardware, but I don't think that's unique. I suppose it might be unique that those blocks can be 64 BYTES WIDE, and so avoid CBC chaining, which means faster ciphering overall, beyond per-block measures.

>>[...]
>>The competition is being conducted in a way which I believe is
>>unconstitutional, which means that the result -- whatever it is --
>>will be open to challenge.
>
>Unconstitutional!? Neat. Which part of the constitution do you see
>being trampled on in this competition?

1) Equal protection under the law.
2) Taking without compensation.
3) Misuse of power; ruling beyond color of law.

You have heard of such things, right? Well you should have -- the first two were in my previous response. Do try to keep up.

>>More than that, these rules act to restrict the long term development
>>of crypto technology by not allowing fundamentally-new technology to
>>compete, and by not rewarding the crypto design process itself. These
>>rules are tools to minimize the open development of cryptographic
>>technology, and every entrant who participates is another government
>>argument that this is a good thing.
>
>As a crypto designer, I have found the whole process more rewarding
>than almost anything else I have done. I expect those rewards to
>continue, even if Twofish does not get chosen.
>
>And honestly, in a standard I would rather see a conservative design
>than fundamentally new technology. If you have the latter, start
>writing papers and getting your ideas into the literature.

If you want articles, I have articles all over my pages. Many have been on-line for years; some are new in the last few months. Since they are available for reading by the public, they are published. Any alleged scientist in cryptography who hasn't kept up with my stuff has nobody to blame but their own lazy self. The major topics are:

1) Dynamic Substitution: keyed, table-based, nonlinear-yet-reversible statistically-balanced combiners for stream ciphers

http://www.io.com/~ritter/#DynSubTech

2) Balanced Block Mixing: arguably perfect balanced reversible mixing, now by keyed nonlinear tables.

http://www.io.com/~ritter/#BBMTech

3) Mixing Ciphers: scalable block ciphers based on (2), supporting toy implementations and huge block sizes which can be dynamically selected (in powers-of-2) at ciphering time. Large blocks avoid the need for CBC chaining and all the sequentiality that implies.

http://www.io.com/~ritter/#MixTech

4) Variable Size Block Ciphers: scalable block ciphers with constant-depth logic and size dynamically-variable to the byte as selected at ciphering time. Modest use of padding in the ciphertext implies that individual blocks cannot even be distinguished, let alone attacked.

http://www.io.com/~ritter/#VSBCTech

Terry Ritter ritter@io.com http://www.io.com/~ritter/
Crypto Glossary 1998-08-27: http://www.io.com/~ritter/GLOSSARY.HTM

Subject: Re: AES and patent rights
Date: Mon, 28 Sep 1998 00:39:39 GMT
From: schneier@counterpane.com (Bruce Schneier)
Message-ID: <360ed738.1402804@news.visi.com>
References: <360ebd79.18691713@news.io.com>
Newsgroups: sci.crypt
Lines: 185

On Sun, 27 Sep 1998 22:34:37 GMT, ritter@io.com (Terry Ritter) wrote:
>On Sat, 26 Sep 1998 21:37:43 GMT, in <360d5983.3024744@news.visi.com>,
>in sci.crypt schneier@counterpane.com (Bruce Schneier) seemed to
>respond, yet failed to address his own analogy:

The analogy was not very good; it wasn't worth defending. I was
thinking of auto racing sponsorship as something done for publicity;
you pointed out that the auto manufacturers got to keep rights to
their designs. Perhaps better would be the sponsorship of a
particular olympics team. When Company X sponsors the U.S. ski team,
they spend money and expertise (or at least, buy expertise) and
receive nothing back except publicity.

>>[...]
>>From what I have been researched, IBM has never sued or even
>>threatened anyone for using DES. If you have other evidence, I very
>>much want to hear it?
>
>Please try to follow along: DES was a US standard. It was free for
>use in the US. Presumably IBM got something for that. Lawsuits and
>threatening have nothing to do with it.

Again, if you know of anything IBM got from DES besides publicity,
please let me know.

>>>My argument would more likely be based on "equal protection under the
>>>law" than antitrust; when government participation is limited by what
>>>one owns, there have to be questions. There is also "taking without
>>>compensation."
>>
>>NIST is not taking anything without compensation. Everything is being
>>given freely. You are not being compelled to submit and to give up
>>your rights.
>
>Indeed, I did not submit.
>
>But *you* get to participate in a government-funded process which took
>nothing from you, but would take property from me.
>
>This is a little more than "not being compelled to submit."

What is AES process taking from you? You were not compelled to
submit, so AES will not take your work away from you. I know that you
patent your ideas, so if the eventual AES algorithm infringes on any
of your patents then you will demand your rights. I don't see
anything of yours being taken away.

>>>[...]
>>>If *I* were in the position where somebody had proposed a system using
>>>my owned technology, I would shut up and wait and hope to be chosen.
>>
>>I know. You and many others.
>
>Life in the property lane: You don't own the land you travel over
>unless you got there first and registered your claim. Deal with it.

I am, as wel all are.

>>All we can do is hope.
>
>Exactly how *I* feel! The whole process could be *quite* amusing.
>
>It's something like a small herd of nearsighted bumbling sheep with
>their own blind shepherd.

Moo. Oops, sorry. Baaa.

>>There is some
>>caselaw on the subject. NIST will make a public call to all third
>>parties to state any potential patent claims regarding the
>>submissions. If someone chooses not to, NIST could argue in court
>>that the patentholder deliberately withheld information in an attempt
>>to hide his rights until after AES was awarded. Will this do any
>>good? No one knows.
>
>As far as I know, there is *no* responsibility in patents to take
>offensive action at any particular time *or* to respond to
>governmental calls for clarification. Perhaps you are thinking of
>copyright.

No, I am thinking of patents. Patentholders must exercise their
rights, or they lose them. In this case, though, I believe you are
correct.

>>[...]
>>>>>It is unfortunate that Bruce Schneier was a prime factor in getting
>>>>>the original rules changed so that only free designs would even be
>>>>>*considered* for AES.
>>>>
>>>>Was I? Wow. I thought that was NIST's idea.
>>>
>>>Oh, please. Are we to believe you have forgotten your letter to NIST
>>>after the first conference? Shall we re-post it?
>>
>>I believe you are confusing my endorsement of an idea with my
>>origination of an idea.
>
>I'm surprised you aren't crying "I don't remember, I can't recall."
>Of course that would do no good, since all this is in the public
>record, but at least it would be more amusing than you blaming me for
>your words.
>
>The NIST position coming out of the first conference was that
>"patented algorithms" (sic) would be accepted, but there would be a
>bias against them. You then argued in a letter -- which you made
>public -- that patented algorithms should not be accepted at all. And
>that became the rule. And of course it was just a coincidence that
>this also stripped off some of your competition.
>
>Are you in charge of NIST? No. Do you bear ultimate responsibility?
>No. Would you be the target of a lawsuit? Probably not. Was your
>letter a major factor in the internal debate? My guess is yes. So,
>are you the proximate cause of the rules change? Probably so.

Well, good for me then. I'm glad I restricted the competition to free
candidates. The last thing we need is another RSADSI-like monopoly.

>>[...]
>>(I'll also pay for a strong bock cipher that encrypt data (ECB mode)
>>in less than 2 clock cycles per byte. I don't know how to do that.)
>
>And once you know, would you still pay?
>
>Certainly my stuff can do 1 cycle PER BLOCK with enough hardware, but
>I don't think that's unique. I suppose it might be unique that those
>blocks can be 64 BYTES WIDE, and so avoid CBC chaining, which means
>faster ciphering overall, beyond per-block measures.

Sorry, I meant on a general-purpose CPU. And I mean a 64-bit codebook
(or a 128-bit codebook).

>>>[...]
>>>The competition is being conducted in a way which I believe is
>>>unconstitutional, which means that the result -- whatever it is --
>>>will be open to challenge.
>>
>>Unconstitutional!? Neat. Which part of the constitution do you see
>>being trampled on in this competition?
>
>1) Equal protection under the law.
>2) Taking without compensation.
>3) Misuse of power; ruling beyond color of law.
>
>You have heard of such things, right? Well you should have -- the
>first two were in my previous response. Do try to keep up.

I'm trying to keep up, but it's hard. I believe that everyone is
being treated equally under the law here. The rules for you are not
different than the rules for me. I don't see anything being taken
without compensation, since the competition is voluntary. And misuse
of power is really stretching things.

I guess I can't keep up. Good luck with your constitutional
challenge.

>>And honestly, in a standard I would rather see a conservative design
>>than fundamentally new technology. If you have the latter, start
>>writing papers and getting your ideas into the literature.
>
>If you want articles, I have articles all over my pages. Many have
>been on-line for years; some are new in the last few months. Since
>they are available for reading by the public, they are published. Any
>alleged scientist in cryptography who hasn't kept up with my stuff has
>nobody to blame but their own lazy self.

Unfortunately, that's not true. (And it is unfortunate.) Publication
does not mean self-publication on a website, it means publication in a
workshop, conference, or journal.

In any case, even if you don't want to publish in conferences or
journals, put cryptanalysis papers on your website. As I said before,
new ideas just for their own sake aren't very interesting. You need
to show how the old ideas are insufficient. You need to break ciphers
designed with the old ideas, and then show how your own ideas are
better.

Designs are dime a dozen, so it's hard to seperate the good ones from
the silly ones. Good cryptanalysis is hard; it will force people to

take notice of your work.

Bruce
****
Bruce Schneier, President, Counterpane Systems Phone: 612-823-1098
101 E Minnehaha Parkway, Minneapolis, MN 55419 Fax: 612-823-1590
Free crypto newsletter. See: http://www.counterpane.com

Subject: Re: AES and patent rights
Date: 28 Sep 1998 03:46:54 GMT
From: stl137@aol.com (STL137)
Message-ID: <19980927234654.06997.00002851@ng111.aol.com>
References: <360ed738.1402804@news.visi.com>
Newsgroups: sci.crypt
Lines: 9

<<Moo. Oops, sorry. Baaa.>>
Har har - this one goes on my quotes page...

STL137@aol.com ===> Website: http://members.aol.com/stl137/
PGP keys: ~~~pgp.html Quotes: ~~~quotes.html
"I have sworn upon the altar of God eternal hostility against every form of
tyranny over the mind of man" - Thomas Jefferson

Subject: Re: AES and patent rights
Date: Mon, 28 Sep 1998 03:57:06 GMT
From: ritter@io.com (Terry Ritter)
Message-ID: <360f0886.6675979@news.io.com>
References: <360ed738.1402804@news.visi.com>
Newsgroups: sci.crypt
Lines: 85

On Mon, 28 Sep 1998 00:39:39 GMT, in <360ed738.1402804@news.visi.com>,
in sci.crypt schneier@counterpane.com (Bruce Schneier) wrote:

>>>[...]
>>>NIST is not taking anything without compensation. Everything is being
>>>given freely. You are not being compelled to submit and to give up
>>>your rights.
>>
>>Indeed, I did not submit.
>>
>>But *you* get to participate in a government-funded process which took
>>nothing from you, but would take property from me.
>>
>>This is a little more than "not being compelled to submit."
>
>What is AES process taking from you? You were not compelled to
>submit, so AES will not take your work away from you. I know that you
>patent your ideas, so if the eventual AES algorithm infringes on any
>of your patents then you will demand your rights. I don't see
>anything of yours being taken away.

Well, this is progress! Now we're about halfway there:

Clearly, if someone else used my work in their submission, I would
"participate" in AES without loss to me. My patents would still apply
even if that design was selected.

But I could not *submit* *my* *own* *work* without loss of rights.

This means that a government process -- one that should apply to me
just the same as you -- would cost me *more* than it cost you. This
is just not equal protection under the law.

>>[...]
>>If you want articles, I have articles all over my pages. Many have
>>been on-line for years; some are new in the last few months. Since
>>they are available for reading by the public, they are published. Any
>>alleged scientist in cryptography who hasn't kept up with my stuff has
>>nobody to blame but their own lazy self.
>
>Unfortunately, that's not true. (And it is unfortunate.) Publication
>does not mean self-publication on a website, it means publication in a
>workshop, conference, or journal.

Fortunately, Science is not the exclusive province of journals, or
even academia. It does not require obeisance from acolytes in
pretentious self-congratulatory symposia nor exist solely in expensive
ink-on-paper prayer books. Science just is.

My stuff is available to anyone. It is not restricted to students who
are physically close to a major library system. It doesn't cost
anything at all, not the price of a symposium, not the price of a
book, not even the price of a CD. In my view, this is the way Science
should be, and that is the way I do it.

The role of a "refereed publication" is (or was useful at one time) to

winnow out some of the chaff and make the resulting journal more worthwhile. This is a *service* to the reader. While this process does define what appears in a particular journal, it is *not* the distinction between Science good and bad. Nor does the term "archival journal" mean what it used to mean.

Academic works generally are *required* to acknowledge the sources of *ideas*, and this is often done even for "private communications" such as personal letters and even private discussions. These are certainly far more questionable than any published works.

Web pages and Usenet articles are published worldwide in form fixed as of a specific date, should have the author's name and a title, and carry both a legal copyright and ethical pride-of-authorship. This is indeed "publication" for academic purposes. Electronic publication can establish legal and moral priority in a field, and is disregarded only by those who wish to be known as academic thieves.

Again, my stuff is available free on my pages. Any alleged scientist in cryptography who hasn't kept up with it has nobody to blame but their own lazy self.

Terry Ritter ritter@io.com http://www.io.com/~ritter/
Crypto Glossary 1998-08-27: http://www.io.com/~ritter/GLOSSARY.HTM

Subject: Re: AES and patent rights
Date: Sun, 27 Sep 1998 21:43:49 -0700
From: Jim Gillogly <jim@acm.org>
Message-ID: <360F1405.ACCBFB83@acm.org>
References: <360f0886.6675979@news.io.com>
Newsgroups: sci.crypt
Lines: 49

> Bruce Schneier wrote:
> >What is AES process taking from you? You were not compelled to
> >submit, so AES will not take your work away from you. I know that you
> >patent your ideas, so if the eventual AES algorithm infringes on any
> >of your patents then you will demand your rights. I don't see
> >anything of yours being taken away.

Terry Ritter wrote:
> Well, this is progress! Now we're about halfway there:
>
> Clearly, if someone else used my work in their submission, I would
> "participate" in AES without loss to me. My patents would still apply
> even if that design was selected.
>
> But I could not *submit* *my* *own* *work* without loss of rights.
>
> This means that a government process -- one that should apply to me
> just the same as you -- would cost me *more* than it cost you. This
> is just not equal protection under the law.

Are you trying to make this into one of those "The rich as well as the poor are prohibited by the law from sleeping under bridges" things? You'll need to spell this one out for me. The way I see it is this:

Bruce & Co. designed and analyzed an algorithm. They submitted it as an AES candidate and chose not to exert any patent rights. They presumably could have patented it if they'd wanted -- the patent office is taking just about anything these days.

You designed and analyzed an algorithm. You patented it. If you had submitted it as an AES candidate you would have had to give up some of those patent rights. You chose not to submit it.

How is this not equal protection under the law? You've simply chosen a different path. Have you yet answered the question about what IBM has received for DES beyond publicity?

You say in another posting that your work has been taken without compensation. Which work? Do one or more of the AES candidates infringe on your patents? What is the extent of the damage to you? Do you think that the process will somehow invalidate your patents?

Feel free to ignore any of these you've already answered -- I stipulate that I have not read all the messages in this thread.

--
Jim Gillogly
Highday, 7 Winterfilth S.R. 1998, 04:27
12.19.5.9.19, 3 Cauac 12 Chen, First Lord of Night

Subject: Re: AES and patent rights
Date: Mon, 28 Sep 1998 08:24:51 GMT
From: ritter@io.com (Terry Ritter)
Message-ID: <360f47cb.22874233@news.io.com>
References: <360F1405.ACCBFB83@acm.org>
Newsgroups: sci.crypt
Lines: 145

On Sun, 27 Sep 1998 21:43:49 -0700, in <360F1405.ACCBFB83@acm.org>, in sci.crypt Jim Gillogly <jim@acm.org> wrote:

>[...]
>Are you trying to make this into one of those "The rich as well as
>the poor are prohibited by the law from sleeping under bridges" things?

I'm not trying to *make* it anything. It is what it is. I am trying to *explain* it, although that does seem rather futile here, everyone being so anxious for their free meal of AES and all. That being the case, what is there to discuss?

>You'll need to spell this one out for me. The way I see it is this:
>
>Bruce & Co. designed and analyzed an algorithm. They submitted it
>as an AES candidate and chose not to exert any patent rights. They
>presumably could have patented it if they'd wanted -- the patent office
>is taking just about anything these days.

Oh, yeah. Why don't *you* just sit down, write a patent, draw figures, construct claims, and prosecute that case through several responses over a period of years to issuance and see how goddamn easy it is? Maybe the process won't seem quite as funny thereafter.

>You designed and analyzed an algorithm. You patented it.

We do not patent algorithms. We can patent implementations of algorithms; they are called "processes." But most of my claims are "machine claims."

>If you had
>submitted it as an AES candidate you would have had to give up some of
>those patent rights. You chose not to submit it.
>
>How is this not equal protection under the law?

OK, one last shot:

Entrant A has no intellectual property to speak of, so he has none to lose. Entering (with the possibility of winning), therefore, is not costly to him.

Entrant B does have intellectual property, established through a complex process of some cost, effort, and time. Entrant B thus has property and investment to lose. Entering, therefore, *is* costly to him.

SO... entering is cheap for A, who has no property, and costly for B, who does.

Why is this hard to understand?

It is true that various interactions with government are based on the property we have: tax rates vary, and welfare and other grants are sometimes means-based. But would it be reasonable to pay more for parking based on the value of one's car? The ideal in the US (which admittedly we never achieve) is that each person has the same vote, and the same worth to the government as any other, independent of property holdings or wealth.

Somewhere there is a dividing line between those services which are provided equally to citizens independent of their means, and those which are not. I claim that AES crossed the line.

>You've simply chosen
>a different path.

Indeed. But I chose this path because the alternative as wrong.

>Have you yet answered the question about what IBM
>has received for DES beyond publicity?

Asking this question makes no sense, and the very fact that it keeps being asked brings into question the motives of those who ask it. It would be just as unreasonable to insist that you show me that IBM received nothing more than publicity. Have you showed that yet?

We do know that intensive negotiation was going on right up until DES was actually published. Negotiation normally means that some form of compromise or reciprocal bargain is being struck. It simply does not take much negotiation to say: "We'll just make this cipher free for use in the US." (There is no question about free use of DES being limited to the US; although AES is to be free *worldwide*.)

There were many ways the government could have provided compensation to the largest computer company in the world, ranging from shifting government contracts to easing export. None of this is necessarily cash on the barrel head, but it would be compensation. From the right

person, a wink and a nod would probably be sufficient. We don't know.

>You say in another posting that your work has been taken without
>compensation.

No, I did not say that, but presumably you took that implication from
what I actually did say.

What I mean is that -- as a condition of AES participation -- rights
had to be given up for the chosen submission. What was actually given
was an *option* on rights, just as one might take an option on land.
But options are themselves worth something. Requiring such an option
as a condition of participation sounds like taking without
compensation to me.

>Which work? Do one or more of the AES candidates
>infringe on your patents?

I doubt it, but I suppose it is possible. Do you have something
particular in mind?

>What is the extent of the damage to you?

Damage?

I suppose the damage here is that I became even more cynical and
disgusted than before by the obvious political manipulations in what
should have been a technical process. Just Washington as usual, I
guess, but that doesn't make it right.

I had been planning to participate in AES for years before AES was
even proposed. You can see work on my pages now about 5 years old
which describes problems with DES and proposes alternatives. I was
one of the first to insist that a 64-bit block was too small. By not
participating I was naturally disappointed, but I feel more knifed in
the back by my government than really caring about the "contest." It
is not a contest.

>Do you think that the process will somehow invalidate your patents?

What? Certainly not. Where did *that* come from?

Terry Ritter ritter@io.com http://www.io.com/~ritter/
Crypto Glossary 1998-08-27: http://www.io.com/~ritter/GLOSSARY.HTM

Subject: Re: AES and patent rights
Date: Mon, 28 Sep 1998 13:35:51 GMT
From: schneier@counterpane.com (Bruce Schneier)
Message-ID: <360f8f29.3658837@news.visi.com>
References: <360f47cb.22874233@news.io.com>
Newsgroups: sci.crypt
Lines: 77

On Mon, 28 Sep 1998 08:24:51 GMT, ritter@io.com (Terry Ritter) wrote:
>OK, one last shot:
>
>Entrant A has no intellectual property to speak of, so he has none to
>lose. Entering (with the possibility of winning), therefore, is not
>costly to him.
>
>Entrant B does have intellectual property, established through a
>complex process of some cost, effort, and time. Entrant B thus has
>property and investment to lose. Entering, therefore, *is* costly to
>him.
>
>SO... entering is cheap for A, who has no property, and costly for B,
>who does.
>
>Why is this hard to understand?

It is hard to understand because you are starting in the middle of the
process. Under your assumptions, A does not have any patent rights
and B does. But we're looking at it one step back, at the idea phase:

Entrant A has an idea. He chooses not to patent it, and instead to
submit it to AES.

Entrant B has an idea. He choosed to patent it, and not to submit it
to AES.

This seems to be the difference here.

>>Have you yet answered the question about what IBM
>>has received for DES beyond publicity?
>
>Asking this question makes no sense, and the very fact that it keeps
>being asked brings into question the motives of those who ask it. It

>would be just as unreasonable to insist that you show me that IBM
>received nothing more than publicity. Have you showed that yet?
>
>We do know that intensive negotiation was going on right up until DES
>was actually published. Negotiation normally means that some form of
>compromise or reciprocal bargain is being struck. It simply does not
>take much negotiation to say: "We'll just make this cipher free for
>use in the US." (There is no question about free use of DES being
>limited to the US; although AES is to be free *worldwide*.)

I suppose he's right. There could have been a secret payoff between NBS and IBM, one that all of DES's designers were kept in the dark about (or which they have lied about all these years). There could be secret payoffs going on to this day between IBM and foreign companies who are using DES.

All we can say is that everyone associated with DES has claimed that IBM gave up all patent right in exchange for nothing, that the official documents agree, and that no one in any country has said that they have paid IBM something for using DES. But Terry is right, this could all be a conspiracy.

>>You say in another posting that your work has been taken without
>>compensation.
>
>No, I did not say that, but presumably you took that implication from
>what I actually did say.
>
>What I mean is that -- as a condition of AES participation -- rights
>had to be given up for the chosen submission. What was actually given
>was an *option* on rights, just as one might take an option on land.
>But options are themselves worth something. Requiring such an option
>as a condition of participation sounds like taking without
>compensation to me.

The difference seems to me that we see that participation is a choice, so there is no taking. He seems to feel differently.

Bruce
****
Bruce Schneier, President, Counterpane Systems Phone: 612-823-1098
101 E Minnehaha Parkway, Minneapolis, MN 55419 Fax: 612-823-1590
Free crypto newsletter. See: http://www.counterpane.com

Subject: Re: AES and patent rights
Date: Mon, 28 Sep 1998 20:36:25 GMT
From: jsavard@tenMAPSONeerf.edmonton.ab.ca (John Savard)
Message-ID: <360ff05c.18633593@news.prosurfr.com>
References: <360f8f29.3658837@news.visi.com>
Newsgroups: sci.crypt
Lines: 33

schneier@counterpane.com (Bruce Schneier) wrote, in part:

>But Terry is right, this
>could all be a conspiracy.

I doubt that Terry Ritter was alleging that sort of conspiracy...

of course, there are other people who claim that DES, IDEA, Blowfish, and all the other well-known block cipher designs are horribly insecure, and suggest that instead we should go and use block ciphers with key-dependent S-boxes with 65,536 entries in them, or Genuine Artificial Imitation One-Time Pads, as the only true road to security.

Obviously, reading your book _Applied Cryptography_ will lead people to suspecting that you are one of the members of this "conspiracy" as well.

As for myself, I'm trying to lead the way to universal security, with designs like the baroque, slow, and hideously complicated Quadibloc II, including a 256-byte key-dependent S-box, the earlier Mishmash proposal which redefines the word "nonlinearity", and my latest large-key brainstorm which rolls together DES, the other Vernam machine (the two-tape system) and the SIGABA, that, on the one hand, are genuinely secure by conventional methods, and yet also include attributes that seem to warm the hearts of those often called "snake-oil vendors", i.e., a huge key, a novel structure that seems bewildering to analyze.

Of course, if I expect incompetent people to use my designs, I really will have to sit down and write some code for them to use...

John Savard
http://members.xoom.com/quadibloc/index.html

Subject: Re: AES and patent rights
Date: Mon, 28 Sep 1998 07:49:18 -0700
From: Jim Gillogly <jim@acm.org>
Message-ID: <360FA1EE.32D300F1@acm.org>
References: <360f47cb.22874233@news.io.com>
Newsgroups: sci.crypt

Lines: 37

I can see we're on different enough wavelengths that doing a
point-by-point argument would be pointless and end up in a
dozen different arguments that don't belong in sci.crypt, but
here's one I can't help objecting to.

> On Sun, 27 Sep 1998 21:43:49 -0700, in <360F1405.ACCBFB83@acm.org>, in
> sci.crypt Jim Gillogly <jim@acm.org> wrote:
> >You designed and analyzed an algorithm. You patented it.

Terry Ritter wrote:
> We do not patent algorithms. We can patent implementations of
> algorithms; they are called "processes."

I recognize that algorithms are not supposed to be patentable, but
in fact a great many algorithms are patented. The patents are
written in stilted and contorted language that casts the algorithm
as a process or machine. In fact this is nonsense, as anyone who's
had the pleasure of reading one of them should be able to attest.
If anybody hasn't, I recommend the LZ77 compression algorithm
patent.

If algorithms couldn't be patented, then programming algorithms
on computers wouldn't violate any patents. Whether you say I've
implemented the process by programming the computer, or you say
I've developed a machine that realizes that patent by programming
a computer, in fact what I have done is implemented an algorithm
by programming the computer.

I don't disagree that designing an important algorithm such as
RSA is a significant piece of intellectual property and the
authors deserve a reward. I do disagree that the patent process
is set up to offer that reward.

--
        Jim Gillogly
        Highday, 7 Winterfilth S.R. 1998, 14:39
        12.19.5.10.0, 4 Ahau 13 Chen, Second Lord of Night

Subject: Re: AES and patent rights
Date: 28 Sep 1998 16:48:50 GMT
From: lamontg@bite.me.spammers
Message-ID: <6uoeli$1amu$1@nntp6.u.washington.edu>
References: <360f47cb.22874233@news.io.com>
Newsgroups: sci.crypt
Lines: 64

ritter@io.com (Terry Ritter) writes:
>>How is this not equal protection under the law?
>
>OK, one last shot:
>
>Entrant A has no intellectual property to speak of, so he has none to
>lose. Entering (with the possibility of winning), therefore, is not
>costly to him.
>
>Entrant B does have intellectual property, established through a
>complex process of some cost, effort, and time. Entrant B thus has
>property and investment to lose. Entering, therefore, *is* costly to
>him.
>
>SO... entering is cheap for A, who has no property, and costly for B,
>who does.
>
>Why is this hard to understand?

Because nobody is forcing you to enter the contest if you judge it to be
too costly to you.

>What I mean is that -- as a condition of AES participation -- rights
>had to be given up for the chosen submission. What was actually given
>was an *option* on rights, just as one might take an option on land.
>But options are themselves worth something. Requiring such an option
>as a condition of participation sounds like taking without
>compensation to me.

Don't participate. Then it isn't taking without compensation. For those
that participate they are entirely free to sign over their rights provided
that they do so without coercion.

Most of your "constitutional argument" is invalidated simply by the fact
that there's nothing illegal or unconstitutional about waiving your rights.

>>What is the extent of the damage to you?
>
>Damage?
>
>I suppose the damage here is that I became even more cynical and
>disgusted than before by the obvious political manipulations in what
>should have been a technical process. Just Washington as usual, I
>guess, but that doesn't make it right.

I'd *love* it if I could sue the government based on my cynicism.

>I had been planning to participate in AES for years before AES was
>even proposed. You can see work on my pages now about 5 years old
>which describes problems with DES and proposes alternatives. I was
>one of the first to insist that a 64-bit block was too small. By not
>participating I was naturally disappointed, but I feel more knifed in
>the back by my government than really caring about the "contest." It
>is not a contest.

It is not a contest that you care to enter, clearly. And you're clearly
upset because you thought you'd get a whole lot of money out of possibly
winning the contest, and you won't. The lottery isn't as rich as you
thought it was going to be, so you're not participating. Here's the
world's smallest violin for you -> .

--
Lamont Granquist (lamontg@u.washington.edu)
looking for unix administration / security work

Subject: Re: AES and patent rights
Date: Tue, 29 Sep 1998 07:34:40 GMT
From: ritter@io.com (Terry Ritter)
Message-ID: <36108d59.28142960@news.io.com>
References: <6uoeli$1amu$1@nntp6.u.washington.edu>
Newsgroups: sci.crypt
Lines: 20

On 28 Sep 1998 16:48:50 GMT, in
<6uoeli$1amu$1@nntp6.u.washington.edu>, in sci.crypt
lamontg@bite.me.spammers wrote:

>[...]
>It is not a contest that you care to enter, clearly. And you're clearly
>upset because you thought you'd get a whole lot of money out of possibly
>winning the contest, and you won't.

Had you been around here when this all started, you would know that I
have never had any delusions about possibly *winning*. I did have the
delusion that I would be able to *participate* without giving away my
second-born child.

Terry Ritter ritter@io.com http://www.io.com/~ritter/
Crypto Glossary 1998-08-27: http://www.io.com/~ritter/GLOSSARY.HTM

Subject: Re: AES and patent rights
Date: Mon, 28 Sep 1998 20:23:14 GMT
From: jsavard@tenMAPSONeerf.edmonton.ab.ca (John Savard)
Message-ID: <360fecd4.17729575@news.prosurfr.com>
References: <360f47cb.22874233@news.io.com>
Newsgroups: sci.crypt
Lines: 53

ritter@io.com (Terry Ritter) wrote, in part:

>Entrant A has no intellectual property to speak of, so he has none to
>lose. Entering (with the possibility of winning), therefore, is not
>costly to him.

>Entrant B does have intellectual property, established through a
>complex process of some cost, effort, and time. Entrant B thus has
>property and investment to lose. Entering, therefore, *is* costly to
>him.

>SO... entering is cheap for A, who has no property, and costly for B,
>who does.

>Why is this hard to understand?

This point does have some validity in the current context, where a
surrender of patent rights is mandatory in the AES. I remember you
making this point earlier, when it was merely "preferred", and there I
found it hard, not to understand, but to believe.

Choosing the lowest bidder (including specifying in advance a maximum
amount one is prepared to pay) is not discrimination; and it looks
like that's what you're trying to argue.

Essentially, NIST is not in the business of awarding free publicity to
whoever has the best cipher algorithm. If that was what AES was about,
discrimination of the type you note would be unfair, since the
patented status of an algorithm is irrelevant to its merit.

What it is instead doing by means of the AES process is: searching for
a good cipher algorithm that the U.S. government can _use in practice_
for safeguarding unclassified but sensitive communications. And a
waiver of royalties with respect to use by parties other than the U.S.
government will further facilitate use of that same algorithm in
communications between the U.S. government and other parties (i.e.,

computerized filing of tax returns).

The free publicity is an incidental consequence of the AES process meeting a goal of the U.S. government, it is not the purpose of the thing from their point of view, however important it may be to the entrants.

However, come to think of it, if we're talking about computerized filing of tax returns, such an application will depend on the use of public-key cryptography, still controlled by patents. Presumably, the main effect of using a royalty-free symmetric algorithm will be to increase the amount the holders of those patents are able to charge while the application remains economically viable. So there is a valid argument for discrimination of a sort...

John Savard
http://members.xoom.com/quadibloc/index.html

Subject: Re: AES and patent rights
Date: Tue, 29 Sep 1998 05:04:06 GMT
From: "Douglas A. Gwyn" <DAGwyn@null.net>
Message-ID: <36106A23.82C0D322@null.net>
References: <360f47cb.22874233@news.io.com>
Newsgroups: sci.crypt
Lines: 11

Terry Ritter wrote:
> There were many ways the government could have provided compensation
> to the largest computer company in the world, ranging from shifting
> government contracts to easing export. None of this is necessarily
> cash on the barrel head, but it would be compensation. From the right
> person, a wink and a nod would probably be sufficient. We don't know.

When government officials get caught engaging in such illegal
procurement activities, the penalties can be severe.
If they didn't need to take that course, there would thus
be considerable incentive not to.

Subject: Re: AES and patent rights
Date: Tue, 29 Sep 1998 19:51:44 -0600
From: jgfunj@EnqvbSerrGrknf.pbz (W T Shaw)
Message-ID: <jgfunj-2909981951440001@dialup163.itexas.net>
References: <36106A23.82C0D322@null.net>
Newsgroups: sci.crypt
Lines: 27

In article <36106A23.82C0D322@null.net>, "Douglas A. Gwyn"
<DAGwyn@null.net> wrote:

> Terry Ritter wrote:
> > There were many ways the government could have provided compensation
> > to the largest computer company in the world, ranging from shifting
> > government contracts to easing export. None of this is necessarily
> > cash on the barrel head, but it would be compensation. From the right
> > person, a wink and a nod would probably be sufficient. We don't know.
>
> When government officials get caught engaging in such illegal
> procurement activities, the penalties can be severe.
> If they didn't need to take that course, there would thus
> be considerable incentive not to.

Are you naive enough to not see what goes on. So much bidding is only
done to present the guise of a credible process. I've seen some of the
worst of this sort of thing. It is not pretty, but almost always gets
hushed up. Under one set of circumstances, the company told me that it
just cost them lots of money to the right places, and it was all my fault
for calling attention to the fradulent activities.
--

Show me a politician who does not lie through his teeth,
and.....I'll show you one who can't find his dentures.

Decrypt with ROT13 to get correct email address.

Subject: Re: AES and patent rights
Date: Mon, 28 Sep 1998 13:25:01 GMT
From: schneier@counterpane.com (Bruce Schneier)
Message-ID: <360f8dcf.3312669@news.visi.com>
References: <360f0886.6675979@news.io.com>
Newsgroups: sci.crypt
Lines: 53

On Mon, 28 Sep 1998 03:57:06 GMT, ritter@io.com (Terry Ritter) wrote:

>
>On Mon, 28 Sep 1998 00:39:39 GMT, in <360ed738.1402804@news.visi.com>,
>in sci.crypt schneier@counterpane.com (Bruce Schneier) wrote:
>
>>>>[...]
>>>>NIST is not taking anything without compensation. Everything is being

>>>>given freely. You are not being compelled to submit and to give up
>>>>your rights.
>>>
>>>Indeed, I did not submit.
>>>
>>>But *you* get to participate in a government-funded process which took
>>>nothing from you, but would take property from me.
>>>
>>>This is a little more than "not being compelled to submit."
>>
>>What is AES process taking from you? You were not compelled to
>>submit, so AES will not take your work away from you. I know that you
>>patent your ideas, so if the eventual AES algorithm infringes on any
>>of your patents then you will demand your rights. I don't see
>>anything of yours being taken away.
>
>Well, this is progress! Now we're about halfway there:
>
>Clearly, if someone else used my work in their submission, I would
>"participate" in AES without loss to me. My patents would still apply
>even if that design was selected.
>
>But I could not *submit* *my* *own* *work* without loss of rights.
>
>This means that a government process -- one that should apply to me
>just the same as you -- would cost me *more* than it cost you. This
>is just not equal protection under the law.

Fascinating.

To me it looks like we were both given the same decision to make, and
you chose one path and I chose the other. You believed that your
patent rights were worth more than NIST was willing to give you for
them. I felt that my patent rights were worth less than the PR NIST
was offering. I believe we were both treated fairly, since were
allowed to make the same decision under the same rules.

But clearly I don't understand constitutional law as well as you do.
Good luck with your suit.

Bruce
****
Bruce Schneier, President, Counterpane Systems Phone: 612-823-1098
101 E Minnehaha Parkway, Minneapolis, MN 55419 Fax: 612-823-1590
Free crypto newsletter. See: http://www.counterpane.com

Subject: Re: AES and patent rights
Date: Tue, 29 Sep 1998 07:38:05 GMT
From: ritter@io.com (Terry Ritter)
Message-ID: <36108e34.28361720@news.io.com>
References: <360f8dcf.3312669@news.visi.com>
Newsgroups: sci.crypt
Lines: 22

On Mon, 28 Sep 1998 13:25:01 GMT, in <360f8dcf.3312669@news.visi.com>,
in sci.crypt schneier@counterpane.com (Bruce Schneier) wrote:

>[...]
>To me it looks like we were both given the same decision to make, and
>you chose one path and I chose the other. You believed that your
>patent rights were worth more than NIST was willing to give you for
>them.

This is sort of a strange comment, isn't it? It might even be the
basis for a sort of occupational joke, where a mathematician gets
"paid" with zero dollars and goes away satisfied! Ha ha, very funny!

Had AES offered even token real compensation for these rights, you
might have a point. They did not.

Terry Ritter ritter@io.com http://www.io.com/~ritter/
Crypto Glossary 1998-08-27: http://www.io.com/~ritter/GLOSSARY.HTM

Subject: Re: AES and patent rights
Date: Tue, 29 Sep 1998 09:52:37 GMT
From: amungedtempdog@munged.see.sig (A [Temporary] Dog)
Message-ID: <3610a518.969320@news.erols.com>
References: <36108e34.28361720@news.io.com>
Newsgroups: sci.crypt
Lines: 38

On Tue, 29 Sep 1998 07:38:05 GMT, ritter@io.com (Terry Ritter) wrote:

>
>On Mon, 28 Sep 1998 13:25:01 GMT, in <360f8dcf.3312669@news.visi.com>,
>in sci.crypt schneier@counterpane.com (Bruce Schneier) wrote:
>
>>[...]

>>To me it looks like we were both given the same decision to make, and
>>you chose one path and I chose the other. You believed that your
>>patent rights were worth more than NIST was willing to give you for
>>them.
>[...]
>Had AES offered even token real compensation for these rights, you
>might have a point. They did not.

If you really believe that the prestige of wining the AES contest is worth nothing, why do you care if you participate or not? If the prestige is worth something (to anyone), it is an offer of compensation. If it's worth nothing, then you have lost nothing by not participating. The AES contestants evidently believe that winning the contest is worth something to them. For some of them, prestige is readily convertible to cash via increased charges for consulting work, etc.

They made an offer (prestige for algorithm). You chose not to accept their offer. Others did choose to accept their offer. This is an example of free trade. The fact that their offer of payment is in intangibles doesn't change that. They didn't force you to participate on their terms and you can't force them to participate on *your* terms. The fact that they are the government and not a business is also irrelevent; it's still an example of free trade.

```
- A (Temporary) Dog            |"There are people who can
The Domain is *erols dot com*  |live and have many diverse
The Name is tempdog            |experiences and learn
                               |nothing" - overheard
Put together as name@domain    |in record store
```

Subject: Re: AES and patent rights
Date: Tue, 29 Sep 1998 18:09:20 GMT
From: ritter@io.com (Terry Ritter)
Message-ID: <361121c4.4707551@news.io.com>
References: <3610a518.969320@news.erols.com>
Newsgroups: sci.crypt
Lines: 87

On Tue, 29 Sep 1998 09:52:37 GMT, in <3610a518.969320@news.erols.com>,
in sci.crypt amungedtempdog@munged.see.sig (A [Temporary] Dog) wrote:

>[...]
>If you really believe that the prestige of wining the AES contest is
>worth nothing,

I do not so believe.

>why do you care if you participate or not?

Well, I was disappointed. But the reason I *care* is that I think it is *wrong*.

>If the
>prestige is worth something (to anyone), it is an offer of
>compensation.

I disagree. This idea of working for prestige demeans the creative element and, indeed, work itself:

* Perhaps writers should write for the privilege of being published.
* Perhaps actors should act for the privilege of being seen.
* Perhaps we all should work for praise alone.

Anyone deserves prestige for the quality of their work. But they also deserve to be compensated for *doing* the work. Here the work is design, and that work deserves compensation. But AES rewards fall to manufacturing, who get a design free. So even though the whole point of this will be to enable a profit-based industry of ciphering hardware and software, there is no profit for the designers. This demeans cipher designers in general, and (what a surprise!) acts to prevent a profit-based industry of cipher design.

>If it's worth nothing, then you have lost nothing by
>not participating.

Some people here have been implying that I lost out by not participating. They may be right. I do not feel that way now.

>The AES contestants evidently believe that winning
>the contest is worth something to them.

I am sure it would be.

>For some of them, prestige is
>readily convertible to cash via increased charges for consulting work,

>etc.

Yes. That would be somewhat more difficult in my case.

>They made an offer (prestige for algorithm).

Silly me, but I would say that it is not NIST that provides prestige, but rather the content of each proposal. In my view, simply being associated with NIST is either of no prestige at all, or is negative.

>You chose not to accept
>their offer. Others did choose to accept their offer. This is an
>example of free trade. The fact that their offer of payment is in
>intangibles doesn't change that.

I believe it does.

>They didn't force you to participate
>on their terms and you can't force them to participate on *your*
>terms. The fact that they are the government and not a business is
>also irrelevent; it's still an example of free trade.

Government is bound by different rules. In AES I believe government stepped over the line.

Terry Ritter ritter@io.com http://www.io.com/~ritter/
Crypto Glossary 1998-08-27: http://www.io.com/~ritter/GLOSSARY.HTM

Subject: Re: AES and patent rights
Date: 29 Sep 1998 14:23:21 -0500
From: juola@mathcs.duq.edu (Patrick Juola)
Message-ID: <6urc39$13i$1@quine.mathcs.duq.edu>
References: <361121c4.4707551@news.io.com>
Newsgroups: sci.crypt
Lines: 44

In article <361121c4.4707551@news.io.com>, Terry Ritter <ritter@io.com> wrote:
>>If the
>>prestige is worth something (to anyone), it is an offer of
>>compensation.
>
>I disagree. This idea of working for prestige demeans the creative
>element and, indeed, work itself:
>
>* Perhaps writers should write for the privilege of being published.
>* Perhaps actors should act for the privilege of being seen.
>* Perhaps we all should work for praise alone.
>
>Anyone deserves prestige for the quality of their work. But they also
>deserve to be compensated for *doing* the work.

Wrong, sir!

Anyone deserves prestige for the quality of their work. If they choose to work for prestige alone, then (by their own choice), that's possibly all the compensation that they get. The world is full of small theatrical productions with amateur actors, putting on plays written by non-professionals, in many cases doing it simply for the love of the work as well as for the exposure and the publicity.

If you're good enough -- and well enough known -- to be able to demand additional compensation for your work, then I congratulate you. But you can't demand that I pay more than I'm willing to pay, especially if there are demonstrably others who are willing to work merely for the praise and exposure.

> Here the work is design, and that work deserves compensation.

And the designers get their compensation in the prestige and exposure from having their candidate algorithms considered.

Perhaps you don't want to work on those terms.

So, don't.

But anyone else who wishes to is free to -- and will reap the appropriate rewards. That's part of the choice they, and you, made, going into the competition (or not).

        -kitten

Subject: Re: AES and patent rights
Date: Tue, 29 Sep 1998 20:38:05 GMT
From: "Douglas A. Gwyn" <DAGwyn@null.net>
Message-ID: <36114507.6025232C@null.net>
References: <361121c4.4707551@news.io.com>

Newsgroups: sci.crypt
Lines: 12

> >They made an offer (prestige for algorithm).
Terry Ritter wrote:
> Silly me, but I would say that it is not NIST that provides prestige,
> but rather the content of each proposal. In my view, simply being
> associated with NIST is either of no prestige at all, or is negative.

Prestige is awarded by others, not by yourself.
Consider:
        (A) "My system was selected as the replacement for DES."
        (B) "I have a better algorithm than AEA."
I think most people are more impressed by the first claim,
which has the merit of being easy to verify.

Subject: Re: AES and patent rights
Date: 30 Sep 1998 03:54:52 GMT
From: jpeschel@aol.com (JPeschel)
Message-ID: <19980929235452.29773.00004008@ng137.aol.com>
References: <361121c4.4707551@news.io.com>
Newsgroups: sci.crypt
Lines: 23

 ritter@io.com (Terry sour grapes Ritter) writes:

>* Perhaps writers should write for the privilege of being published.
>* Perhaps actors should act for the privilege of being seen.
>* Perhaps we all should work for praise alone.
>
Crazy man -- can you dig it! When do we do zap the capitalist system
into a blue way-gone haze?

Jack Kerouac

oh yeah, Joe says -- go here, too...cool...

_____

Joe Peschel
D.O.E. SysWorks
http://members.aol.com/jpeschel/index.htm
_____

Subject: Re: AES and patent rights
Date: Wed, 30 Sep 1998 14:55:05 +0900
From: "Lenz" <lenz@als.aoyama.ac.jp>
Message-ID: <6ushf2$lao@enews2.newsguy.com>
References: <361121c4.4707551@news.io.com>
Newsgroups: sci.crypt
Lines: 30

Terry Ritter wrote in message <361121c4.4707551@news.io.com>...

>Government is bound by different rules. In AES I believe government
>stepped over the line.

Is there a rule in American law which prevents NIST from doing what they
did? If so, in what law? Looking at an article at lawlinks.com/beck.html my
impression is that both options are possible. Certainly the people at NIST
should know about any restricting rule if there is one, since the patent
question seems to be important in just about any standardization process.

If a government standard adopts patented technology, that would be
government influence on the market competition. The job of the government is
to stop restrictions of market competition, not to take part in them. So I
think that a government standard should avoid patented technology.

Any compensation for cipher design needs to come from winning in the
marketplace, not in AES. Since many good ciphers are available for free,
anyone wanting to charge for cipher design needs to beat major league teams
as NTT and IBM, as well as many other serious players. Not participating in
AES does not stop anyone from trying to do so. That means that any lack of
compensation is just an indication of failure in the marketplace, which you
probably would not want to claim for your designs.

Karl-Friedrich Lenz :-)
www.toptext.com/crypto/

Subject: Re: AES and patent rights
Date: Wed, 30 Sep 1998 08:11:26 GMT
From: bryanolson@my-dejanews.com

Message-ID: <6usp3c$o3l$1@nnrp1.dejanews.com>
References: <361121c4.4707551@news.io.com>
Newsgroups: sci.crypt
Lines: 22

Terry Ritter wrote:

> Anyone deserves prestige for the quality of their work. But they also
> deserve to be compensated for *doing* the work. Here the work is
> design, and that work deserves compensation.

Actually we pay what we have to, to get what we want.
No one advertises by saying he deserves the money. NIST
correctly predicted that the worlds best cryptographers
would offer top quality designs for free. It's not a
government decree that pushed the monetary price of
symmetric ciphers to zero; it's competition.

Very few people were paying for symmetric ciphers before
the AES announcement. The market had spoken and NIST
listened.

--Bryan

-----== Posted via Deja News, The Leader in Internet Discussion ==-----
http://www.dejanews.com/rg_mkgrp.xp Create Your Own Free Member Forum

---

Subject: Re: AES and patent rights
Date: Wed, 30 Sep 1998 14:25:58 GMT
From: nospam@pd.jaring.my (Lincoln Yeoh)
Message-ID: <36123b24.4347337@nntp.jaring.my>
References: <361121c4.4707551@news.io.com>
Newsgroups: sci.crypt
Lines: 36

On Tue, 29 Sep 1998 18:09:20 GMT, ritter@io.com (Terry Ritter) wrote:
>
>>They didn't force you to participate
>>on their terms and you can't force them to participate on *your*
>>terms. The fact that they are the government and not a business is
>>also irrelevent; it's still an example of free trade.
>
>Government is bound by different rules. In AES I believe government
>stepped over the line.
>

I see AES as a crypto donation drive- "Donate Decent Crypto to the World".
You can't give and keep at the same time. If you want to keep, you can't
qualify.

You are still free to charge for your crypto. It won't be the world famous
free AES but it'll still be yours and you'll get to keep it.

I don't see how you have been wronged. How has the Gov stepped over the
line?

Hmm, I haven't complained to the blood donating organisers that it is wrong
to tell willing donors beforehand that if their blood is suitable they have
to give it for free.

What would be wrong if they purposely selected bad or flawed crypto. That
is my worry - how would we know? Perhaps the NSA have figured out the flaws
in all the free stuff out there? Pure speculation of course.

Link.

****
Reply to: @Spam to
lyeoh at @people@uu.net
pop.jaring.my @
****

Subject: Re: AES and patent rights
Date: Wed, 30 Sep 1998 20:39:55 -0500
From: "Stephen M. Gardner" <gardner@metronet.com>
Message-ID: <3612DD6B.93586E63@metronet.com>
References: <361121c4.4707551@news.io.com>
Newsgroups: sci.crypt
Lines: 103

Terry Ritter wrote:
> I disagree. This idea of working for prestige demeans the creative
> element and, indeed, work itself:
I'm really amazed at this statement. The vast majority of creative
people outside of entrepreneurial engineering sorts would think you have
it completely upside down. Many would say that when you do something
just for money you cheapen it. After all, when you have the talent it
just has to come out whether someone is willing to pay or not. Many
artists work for the simple joy of creating. You really have to talk to
more artists and actors. The really good ones live for their art and do
what they can to survive. Often they are not "rewarded" until after

death. There is a great line in an old Charles Aznavour song "La Boheme" that talks about this:

Dans les cafes voisins, nous etions quelques uns qui attendions la gloire.
Et, bien que misereux, avec le ventre creux, nous ne cessions d'y croire.
Et, quand quelque bistro, contre un bon repas chaud nous prenait une toile.
Nous recitions des vers groupe autours du poele en oubliant l'hiver. . .

In the neighborhood cafes, there were a few of us expecting glory.
And, even though we were destitute, with our bellies empty, we never stopped
believing.
And when a bistro took one of our canvases in exchange for a hot meal.
We sat around the stove reciting poetry, forgetting the winter. . .

Perhaps I'm a hopeless romantic but I find this easy to understand. In fact, I can get downright misty eyed about this song if it is cold out and I'm drinking a nice French red. ;-) And it's not just artists either. Don't forget that Einstein never made a lot of money off of his amazing creativity. I think he would have considered it crass and bourgeois. He did his best work while working at a deadly boring job in the Swiss Patent Office. He didn't get a red cent for "On the Electrodynamics of Moving Bodies" but it made him the most famous man in the world a few years later. I don't think he would understand the late 20th century Texas entrepreneur very well. ;-)

>Perhaps writers should write for the privilege of being published.
They often do until they are discovered.

>Perhaps actors should act for the privilege of being seen.
Does Summer Stock mean anything to you?

>Perhaps we all should work for praise alone.
Terry, I'm sorry but I think you sound petulant and spoiled here. Giving a single algorithm to the public domain could have had great benefits. It makes you sound very spoiled and childish to be complaining this way. I bet Linus Torvalds isn't sorry that Linux is in the public domain. There is an ol' boy that could get a job anywhere in the world.

> Anyone deserves prestige for the quality of their work. But they also
> deserve to be compensated for *doing* the work. Here the work is
> design, and that work deserves compensation.
You need to talk to more creative people outside of engineering. Actors, writers, poets, painters, sculptors they all do a lot of their best work for pennies or nothing at all until they get famous.

> Some people here have been implying that I lost out by not
> participating. They may be right. I do not feel that way now.
Then you should be happy. You didn't get taken in by that facile fraud. ;-)

> Silly me, but I would say that it is not NIST that provides prestige,
> but rather the content of each proposal. In my view, simply being
> associated with NIST is either of no prestige at all, or is negative.
Perhaps some of your failure to thrive economically is due to a misunderstanding of your market. I think most of your potential customers would beg to differ here. And here is a little fact for you about capitalism: it doesn't matter one whit whether your customers are right in their appreciation of the NIST or not. If they have money, want tons of shit and you have shit to sell then you are a good business man. I think perhaps you have a hard time figuring out whether you want to be a business man or an "artiste". ;-) Charles Aznavour might understand. ;-)

> >They didn't force you to participate
> >on their terms and you can't force them to participate on *your*
> >terms. The fact that they are the government and not a business is
> >also irrelevent; it's still an example of free trade.
>
> Government is bound by different rules. In AES I believe government
> stepped over the line.
That is your perogative. The nice thing about living in relative freedom is that you get to believe that about government without being dragged off in the night. I happen to believe differently in this particular point. And sadly for you I think your potential customers are on a different wavelength too.

--
Take a walk on the wild side: http://www.metronet.com/~gardner/
Still a lot of lands to see but I wouldn't want to stay here,
it's too old and cold and settled in its ways here.
Joni Mitchell ("California")

Subject: Re: AES and patent rights
Date: 1 Oct 1998 13:43:58 GMT
From: mdw@catbert.ebi.ac.uk (Mark Wooding)
Message-ID: <slrn7171ot.8vf.mdw@catbert.ebi.ac.uk>
References: <361121c4.4707551@news.io.com>
Newsgroups: sci.crypt
Lines: 25

Terry Ritter <ritter@io.com> wrote:

> I disagree. This idea of working for prestige demeans the creative
> element and, indeed, work itself:
>
> * Perhaps writers should write for the privilege of being published.
> * Perhaps actors should act for the privilege of being seen.
> * Perhaps we all should work for praise alone.
>
> Anyone deserves prestige for the quality of their work. But they also
> deserve to be compensated for *doing* the work.

I do lots of things I enjoy. One of them is writing software. Another is administrating Unix systems. Yet another is writing low-quality poetry.

If I could, I'd do them all for the joy of it. It's a cause of annoyance to me that this isn't possible. So I have to do a job in order to have enough money to do the things I like doing. Lucky me: my job involves doing at least one of the things I like doing anyway.

Creativity exists because, fundamentally, people enjoy being creative. That ought to be enough. It's a real shame it isn't.

-- [mdw]

Subject: Re: AES and patent rights
Date: Thu, 01 Oct 1998 11:28:33 -0600
From: jgfunj@EnqvbSerrGrknf.pbz (W T Shaw)
Message-ID: <jgfunj-0110981128330001@dialup81.itexas.net>
References: <slrn7171ot.8vf.mdw@catbert.ebi.ac.uk>
Newsgroups: sci.crypt
Lines: 32

In article <slrn7171ot.8vf.mdw@catbert.ebi.ac.uk>, mdw@ebi.ac.uk wrote:
>
> Creativity exists because, fundamentally, people enjoy being creative.
> That ought to be enough. It's a real shame it isn't.
>
Perhaps people who can be especially creative in a particular field that produces worthwhile technology should be supported by those who are not particularily creative themselves. Boy, that sound's like lots of *commercial* enterprises.

If patents are truely justified, they should simply be issued and not need to be serviced. To do otherwise is to punish and discourage individual initiative. Too often this is simply done to keep upstarts from entering the market, rather an unAmerican thing to do. Consider the willing involvement by big buisiness for targeted subversion of the intellectual property area.

When it comes to crypto, some other areas too, things are twisted upside down from the classic, nice talk about free enterprise, so the government becomes the fixer, manipulator, and irreverant boss who ultimately selects blesses the commercial endeavors that submit to the rules. This is an old and not too pretty pattern that is hard to defeat out of habit.

Cut the payola, the bribes, which is a Constitutional sanctioned trigger for impeachment of elective and certain appointed governmental officials, and the processes would be cleaner. So much else is mere cookiework.
--

Show me a politician who does not lie through his teeth,
and.....I'll show you one who can't find his dentures.

Decrypt with ROT13 to get correct email address.

Subject: Re: AES and patent rights
Date: Tue, 29 Sep 1998 19:49:22 GMT
From: "Douglas A. Gwyn" <DAGwyn@null.net>
Message-ID: <3611399D.9643C097@null.net>
References: <3610a518.969320@news.erols.com>
Newsgroups: sci.crypt
Lines: 14

A [Temporary] Dog wrote:
> If you really believe that the prestige of wining the AES contest is
> worth nothing, why do you care if you participate or not?

I think a key (tacit) element in Terry's reasoning is that AES will be widely used, even mandatory in some cases, displacing commercial systems that might have been used instead.

My immediate response is that without AES, we'd be seeing

either 3DES or some NSA ISSO-devised system instead,
which would still be free etc.

If AES had resulted in a mandatory-federal-use commercial
product, now *that* might reasonably be viewed as unfair.

Subject: Re: AES and patent rights
Date: Wed, 30 Sep 1998 13:35:51 GMT
From: "Joseph K. Nilaad" <jknilaad@ssd.bna.boeing.com>
Message-ID: <361233B7.5E20@ssd.bna.boeing.com>
References: <3610a518.969320@news.erols.com>
Newsgroups: sci.crypt
Lines: 32

A [Temporary] Dog wrote:
>
> If you really believe that the prestige of wining the AES contest is
> worth nothing, why do you care if you participate or not? If the
> prestige is worth something (to anyone), it is an offer of
> compensation. If it's worth nothing, then you have lost nothing by
> not participating. The AES contestants evidently believe that winning
> the contest is worth something to them. For some of them, prestige is
> readily convertible to cash via increased charges for consulting work,
> etc.
>
> They made an offer (prestige for algorithm). You chose not to accept
> their offer. Others did choose to accept their offer. This is an
> example of free trade. The fact that their offer of payment is in
> intangibles doesn't change that. They didn't force you to participate
> on their terms and you can't force them to participate on *your*
> terms. The fact that they are the government and not a business is
> also irrelevent; it's still an example of free trade.
>
I see. What if Boston marathon gives only the trophy and no other
incentives, I wonder how many world class runners would participate?
I do agree with you that nobody forces anybody to participate, but if
you put a very strict constraints to your requirements, there may not be
enough contestants to determine which is the best. If Boston marathon
committee say that only people who live in Boston and never
win any marathon before can participate, then how many people will
participate? Probably, there would be just a bunch of amateur entries.

I don't think we're all working for charity!

Subject: Re: AES and patent rights
Date: Wed, 30 Sep 1998 10:12:40 -0500
From: "R H Braddam" <rbraddam@aic-fl.com>
Message-ID: <6uthqg$5an$1@server.cntfl.com>
References: <361233B7.5E20@ssd.bna.boeing.com>
Newsgroups: sci.crypt
Lines: 114

Joseph K. Nilaad wrote in message <361233B7.5E20@ssd.bna.boeing.com>...
>I see. What if Boston marathon gives only the trophy and no other
>incentives, I wonder how many world class runners would participate?
>I do agree with you that nobody forces anybody to participate, but if
>you put a very strict constraints to your requirements, there may not be
>enough contestants to determine which is the best. If Boston marathon
>committee say that only people who live in Boston and never
>win any marathon before can participate, then how many people will
>participate? Probably, there would be just a bunch of amateur entries.

Being amateur doesn't necessarily mean not being good. Olympic atheletes
train for years to be the best at what they do. Some of them do it for the
self-satisfaction they get from their accomplishments.

>I don't think we're all working for charity!

It may be more a matter of competition with their peers to "build the better
mousetrap". And just because the encryption algorithm is free don't mean
that the applications using them will be. Who will be the established expert
on the algorithm which is chosen? Who will be in the best position to assist
program developers with incorporating the algorithm into their products?

I have a question for the group about the AES entrants. How many of them
were public domain before being submitted to the AES?

Ritter suggests that those who entered their algorithms gave up nothing,
that their property had no value. If their entries were secret until
submitted, they could have patented them and licensed them, and therefore
may have lost as much through their donations as anyone would have lost who
had patented algorithms. And if they will reap benefits through their
business as a result of their donations. So could have anyone else who
donated their patented work if their algorithms were chosen as the standard.

I don't intend this to be critical of Terry Ritter. I visited his web site,
and saved his pages to disk so I can go back to them and study them offline.

I can work my way through the code for Blowfish, SHA1, et al, but I can read and understand Ritter's description of his methods much quicker. They seem to make sense, and I'm sure they weren't developed overnight. I understand why a patent is important to him, to protect years of research and work. I think he has a
valid complaint, it is just that he over simplified his analogy. I also agree with Bruce that no one is compelled (by outside forces) to donate their algorithms to the public domain via the AES.

I also believe that a public domain standard will result in the incorporation of cryptography into products much quicker than a licensed standard would. Sure, large software houses can pay "fair and non-discriminatory" licensing fees easily, but what is fair for a large business is often out of reach for an independent programmer. And far more applications software, often of very high quality, comes from independents than from the large businesses.

I don't believe that adoption of a public domain AES will reduce anyone's ability to derive income from their patented crypto technology. There is plenty of room for more than one way to do things here. Also, the more choices available for individual users, the harder for an attacker to keep up. The business market is very large, with millions of transactions daily. There aren't that many users, though, when you consider that they all have to communicate with each other. They have to talk the same cryptography.

The personal market doesn't have the same requirements or restrictions. There are potentially thousands of times more personal users than businesses. The issue for them is privacy, and they are going to demand it when they realize how open their communications are. People from all nations around the world will fight and die for freedom, and privacy is an essential part of freedom.

It would seem to me that the way to get widespread acceptance of a particular method would be to get it into as many products as possible, as quickly as possible. No method or product can capture a market share without being on the market. Often, the first of a new type of product becomes the leader and sets the standards for those that follow.

Eventually, encryption of the operating system itself may be considered necessary, to prevent damage to systems which are on-line 100% of the time. That may require a completely different way of looking at cryptography. I expect to see cryptographic software combined with existing hardware in new systems. Many systems can now be configured to require a password before they even begin to boot the operating system. Combine that with a mag stripe or barcode reader for credit card sized plastic cards, and you have a good user login system.

Anyone could carry a notarized signature around by carrying a card with a certificate written on the mag stripe by an institution (bank) or branch of the government. Businesses could issue their own cards for access control to their network from anywhere in the world over the internet. Smart cards may be a great idea, but credit cards are here now, widely accepted, and inexpensive to field.

It takes more than a good (or great) algorithm to achieve commercial success. It has to be easily available to the public, inexpensive, and easy to install. Once installed, it has to be practically invisible to the user.

Another need is for an SSL filter dll for Personal Web Server. PWS is widely available, for free, for Windows 95 and 98. It has adequate capacity for small businesses, and handles ActiveX and Active Server Pages. Eighty percent of the businesses in the US are small businesses. Many of them have a large regional and even national market area. They have sales reps and distributors nationwide. But they won't pay the price to move to a large and complicated to maintain operating system like NT or UNIX. Many of them won't even pay for an extra machine to use as a server for Novell for their LAN. They CAN be brought into the 21st century if there exists an entry-level way to do it, then they'll move up to larger systems when they find out that it will work for them.

Sorry this got so long. I hope someone takes the time to read it.
Rickey

Subject: Re: AES and patent rights
Date: Wed, 30 Sep 1998 19:15:13 GMT
From: schneier@counterpane.com (Bruce Schneier)
Message-ID: <36128239.16547455@news.visi.com>
References: <6uthqg$5an$1@server.cntfl.com>
Newsgroups: sci.crypt
Lines: 38

On Wed, 30 Sep 1998 10:12:40 -0500, "R H Braddam"
<rbraddam@aic-fl.com> wrote:
>I have a question for the group about the AES entrants. How many of them
>were public domain before being submitted to the AES?

None were, since they were all new. But several entries have been put
in the public domain at the time of submitting, regardless of whether
they win or lose.

Twofish, Serpent, Rijndael, LOKI-97, HPC, DEAL, and SAFER+ are in the public domain right now.

RC6, Mars, and DFC are not in the public domain right now.

CAST-256, E2, Frog, Magenta, and Crypton I don't know about. CAST-128 is in the public domain (without being any standard), so CAST-256 may be as well. E2 is from NTT, so it is probably not in the public domain. Crypton is by a Korean academic, and is probably in the public domain. Magenta is by Deutche Telcom, and is probably not. Frog...well the designers are on this newsgroup and can comment for themselves.

>Ritter suggests that those who entered their algorithms gave up nothing,
>that their property had no value. If their entries were secret until
>submitted, they could have patented them and licensed them, and therefore
>may have lost as much through their donations as anyone would have lost who
>had patented algorithms. And if they will reap benefits through their
>business as a result of their donations. So could have anyone else who
>donated their patented work if their algorithms were chosen as the standard.

Presumably some of the AES submissions have patent applications pending.

Bruce
****
Bruce Schneier, President, Counterpane Systems Phone: 612-823-1098
101 E Minnehaha Parkway, Minneapolis, MN 55419 Fax: 612-823-1590
Free crypto newsletter. See: http://www.counterpane.com

Subject: Re: AES and patent rights
Date: Thu, 01 Oct 1998 05:38:11 GMT
From: dianelos@tecapro.com
Message-ID: <6uv4g3$f5g$1@nnrp1.dejanews.com>
References: <36128239.16547455@news.visi.com>
Newsgroups: sci.crypt
Lines: 19

In article <36128239.16547455@news.visi.com>,
schneier@counterpane.com (Bruce Schneier) wrote:
>[...]
> CAST-256, E2, Frog, Magenta, and Crypton I don't know about. CAST-128
> is in the public domain (without being any standard), so CAST-256 may
> be as well. E2 is from NTT, so it is probably not in the public
> domain. Crypton is by a Korean academic, and is probably in the
> public domain. Magenta is by Deutche Telcom, and is probably not.
> Frog...well the designers are on this newsgroup and can comment for
> themselves.

FROG is in the public domain too.

--
http://www.tecapro.com
email: dianelos@tecapro.com

----------== Posted via Deja News, The Discussion Network ==----------
http://www.dejanews.com/ Search, Read, Discuss, or Start Your Own

---

Subject: Re: AES and patent rights
Date: 30 Sep 1998 11:05:52 -0500
From: juola@mathcs.duq.edu (Patrick Juola)
Message-ID: <6utkt0$2p7$1@quine.mathcs.duq.edu>
References: <361233B7.5E20@ssd.bna.boeing.com>
Newsgroups: sci.crypt
Lines: 29

In article <361233B7.5E20@ssd.bna.boeing.com>,
Joseph K. Nilaad <jknilaad@ssd.bna.boeing.com> wrote:
>A [Temporary] Dog wrote:
>>
>> If you really believe that the prestige of wining the AES contest is
>> worth nothing, why do you care if you participate or not? If the
>> prestige is worth something (to anyone), it is an offer of
>> compensation. If it's worth nothing, then you have lost nothing by
>> not participating. The AES contestants evidently believe that winning
>> the contest is worth something to them. For some of them, prestige is
>> readily convertible to cash via increased charges for consulting work,
>> etc.
>>
>> They made an offer (prestige for algorithm). You chose not to accept
>> their offer. Others did choose to accept their offer. This is an
>> example of free trade. The fact that their offer of payment is in
>> intangibles doesn't change that. They didn't force you to participate
>> on their terms and you can't force them to participate on *your*
>> terms. The fact that they are the government and not a business is
>> also irrelevent; it's still an example of free trade.
>>
>I see. What if Boston marathon gives only the trophy and no other
>incentives, I wonder how many world class runners would participate?

Probably a hell of a lot. The Olympics certainly pay "only the medal," yet have no problem getting competitors.

-kitten

Subject: Re: AES and patent rights
Date: Sat, 03 Oct 1998 14:24:11 GMT
From: nospam@pd.jaring.my (Lincoln Yeoh)
Message-ID: <3616314d.2983645@nntp.jaring.my>
References: <3612a746.1207056@news.erols.com>
<361233B7.5E20@ssd.bna.boeing.com>
Newsgroups: sci.crypt
Lines: 36

On Wed, 30 Sep 1998 22:59:46 GMT, amungedtempdog@munged.see.sig (A [Temporary] Dog) wrote:

>My post wasn't really intended to address the question of whether it
>was wise to restrict entry to public domain algorithms. My main point
>was that an intangible reward is still a reward, and can still be a
>basis for trade.

Heh the only intangible reward is one which isn't a reward in the first place.

Patpat, good boy! ;).

Of course most people don't get paid in pats on backs, praises, respect etc, but it actually matters a lot to many people. But in an increasingly monetary minded society, people tend to lose sight of how much they might enjoy such things.

It ain't just money, if it was, well then it really is stupid. Money/wealth is just a concept which most people agree to recognise. Society agrees that if A has a bigger number than B in the net worth field, A is richer.

There are many other worthwhile concepts that people should not forget to recognise. If I do something for a friend as an act of friendship, then I don't expect payment, in fact it kind of spoils things.

Pity the High Priests of the Free Market seem to have a louder voice nowadays, I'm not even sure of what their real doctrines are anymore, given what they say.

Link.
****
Reply to: @Spam to
lyeoh at @people@uu.net
pop.jaring.my @
****

Subject: Re: AES and patent rights
Date: Wed, 30 Sep 1998 18:59:32 GMT
From: "Joseph K. Nilaad" <jknilaad@ssd.bna.boeing.com>
Message-ID: <36127F94.5955@ssd.bna.boeing.com>
References: <3610a518.969320@news.erols.com>
Newsgroups: sci.crypt
Lines: 75

R H Braddam wrote:
>
> Being amateur doesn't necessarily mean not being good. Olympic atheletes
> train for years to be the best at what they do. Some of them do it for the
> self-satisfaction they get from their accomplishments.
Olympic? Hmm. Why did I see Magic Johnson, Michael Jordan, Graff, etc.
in
Bacelona, Spain? I thought the whole idea of Olympic atheletes is to
bring
out the *best* atheletes in each field from each country. Thanks for
bringing
Olympic scenario up, who is collecting the money? Certainly not the
atheletes.
I have not yet seen any free admission in Olympic.

> mousetrap". And just because the encryption algorithm is free don't mean
> that the applications using them will be. Who will be the established expert
I agree. But what is bugging me is that AES to be "free world wide".
That
is a joke unless NIST can change the export law (40-bit), can't they? I
had
asked Bruce, himself couldn't give a definite answer what does "free
world
wide" really mean. The major concern is the algorithm. "Free world
wide"
sounds pretty good mousetrap.

> Ritter suggests that those who entered their algorithms gave up nothing,
> that their property had no value. If their entries were secret until
> submitted, they could have patented them and licensed them, and therefore
If something is patented, it is not secret. Using it without permission
from

the invertor that is a no no. It should be up to the inventor he/she wants
to charge for use, not somebody telling him/her what to do. By the way, did
IBM give up their rights to DES?

> why a patent is important to him, to protect years of research and work. I
> think he has a
> valid complaint, it is just that he over simplified his analogy. I also
> agree with Bruce that no one is compelled (by outside forces) to donate
> their algorithms to the public domain via the AES.
I understand both point of views.

> business is often out of reach for an independent programmer. And far more
> applications software, often of very high quality, comes from independents
> than from the large businesses.
I agree totally, eg., Alan Cooper, Fox Software, etc.

> The personal market doesn't have the same requirements or restrictions.
> There are potentially thousands of times more personal users than
> businesses. The issue for them is privacy, and they are going to demand it
> when they realize how open their communications are. People from all nations
> around the world will fight and die for freedom, and privacy is an essential
> part of freedom.
>
> It would seem to me that the way to get widespread acceptance of a
> particular method would be to get it into as many products as possible, as
> quickly as possible. No method or product can capture a market share without
> being on the market. Often, the first of a new type of product becomes the
> leader and sets the standards for those that follow.
I see some potential problem here. See my second reply.

> Sorry this got so long. I hope someone takes the time to read it.
That's alright. We haven't had a good thread for a while now.

Joe Nilaad
Would that be great if we have a cryptosystem that could survive the test,
providing that the adversary has complete plaintext and complete ciphers.
I wonder...

Subject: Re: AES and patent rights
Date: Thu, 1 Oct 1998 21:25:27 -0500
From: "R H Braddam" <rbraddam@aic-fl.com>
Message-ID: <6v1dipojk1@server.cntfl.com>
References: <36127F94.5955@ssd.bna.boeing.com>
Newsgroups: sci.crypt
Lines: 34

Joseph K. Nilaad wrote in message <36127F94.5955@ssd.bna.boeing.com>...
>I thought the whole idea of Olympic athletes is to
>bring
>out the *best* athletes in each field from each country. Thanks for

Haven't Olympic champions been stripped of their awards because of professional/commercial activity, even if it was miniscule or accidental??

>
>If something is patented, it is not secret. Using it without permission
>from

I meant, if their entries were not in the public domain before they submitted them to the AES, they could patent them instead of submitting them to the AES. Until they had released the secret, they could patent their algorithm. Once they release the secret, they may have difficulty getting a patent or enforcing one. However, if they obtained a patent, then the AES came up, then they would have had to make the same decision under the came conditions as faced Mr.. Ritter. They would have also had to consider the potential loss of income from sales or licensing, same as Mr.. Ritter. Therefore, just because the other submissions were not YET patented, that doesn't mean they were unpatentable, or worthless.
>
>I see some potential problem here. See my second reply.

Agreed. However, the times, they are a-changing, and so will the export laws. Also, there is a major market in the U.S. alone, and software for the US market can be written elsewhere and exported to the US. Get things started here, and watch them spread. BTW, Windows bashing may be fun, but there (in Windows products) exists a MAJOR market.

Subject: Re: AES and patent rights
Date: Fri, 2 Oct 1998 14:04:57 GMT
From: "Joseph K. Nilaad" <jknilaad@ssd.bna.boeing.com>
Message-ID: <3614DD89.4F95@ssd.bna.boeing.com>
References: <6v1dipojk1@server.cntfl.com>
Newsgroups: sci.crypt

Lines: 52

R H Braddam wrote:
>
> I meant, if their entries were not in the public domain before they
> submitted them to the AES, they could patent them instead of submitting them
> to the AES. Until they had released the secret, they could patent their
> algorithm. Once they release the secret, they may have difficulty getting a
> patent or enforcing one. However, if they obtained a patent, then the AES
> came up, then they would have had to make the same decision under the came
> conditions as faced Mr.. Ritter. They would have also had to consider the
> potential loss of income from sales or licensing, same as Mr.. Ritter.
> Therefore, just because the other submissions were not YET patented, that
> doesn't mean they were unpatentable, or worthless.
Why don't AES just simply state that PATENT *PENDING* ALGORITHM IS NOT
ACCEPTABLE. Then for those who had patented algorithm can enter and Mr.
Ritter
can rest his case. Those guys who have patent pendings and wish to
enter
AES contest, they have to make decision whether they want to relinquish
the
rights on their algorithm or not, in which they know darn well that in
crypto society, THE ALGORITHM MUST BE OPENED!

> Agreed. However, the times, they are a-changing, and so will the export
> laws. Also, there is a major market in the U.S. alone, and software for the
Changing? That would take the act of God. As big as beuracrats is, I
don't
see it in anytime in near future. I mean like less than 5 years. This
may
not include time to determine AES winner.

Free worldwide and 40bit export law at this moment and in this country,
is a
contradiction! For those who jump in "free worldwide" band wagon, I
don't
know what are they thinking. I am sure they all have good deed, but
they
just might have forgot some other issue.

Sorry world, you have to wait.

> started here, and watch them spread. BTW, Windows bashing may be fun, but
> there (in Windows products) exists a MAJOR market.
Boy boy do I agree with you on this. As long as people keep spending
money on
beta test units and keep debugging for WHEELIE BILLY, the market will
always
exist. It seems to me that writing buggy program is good business so
that the
sellers can say "Oh, you'll have to wait or upgrade the next
version/release"
I do use MS products, but not by choice.

Subject: Re: AES and patent rights
Date: Fri, 2 Oct 1998 14:06:31 -0500
From: "R H Braddam" <rbraddam@aic-fl.com>
Message-ID: <6v387p$65l$1@server.cntfl.com>
References: <3614DD89.4F95@ssd.bna.boeing.com>
Newsgroups: sci.crypt
Lines: 57

Joseph K. Nilaad wrote in message <3614DD89.4F95@ssd.bna.boeing.com>...
-- snipped my remarks --

>Changing? That would take the act of God. As big as beuracrats is, I
>don't see it in anytime in near future. I mean like less than 5 years.
This
>may not include time to determine AES winner.

Yes, changing. The original ITAR export controls were very likely
unconstitutional, and the movement of control to Department of Commerce was
just a stop-gap measure to delay or circumvent having all or a majority of
them thrown out by the Supreme Court. Recent government announcements of
reduction of export controls, and elimination of some (for banking &
insurance) is the start of a down hill slide. The more they do, the more
they attract the attention of the general public, and the more pressure they
will get to further de-control crypto.

>Free worldwide and 40bit export law at this moment and in this country,
>is a contradiction! For those who jump in "free worldwide" band wagon, I
>don't know what are they thinking. I am sure they all have good deed, but
>they just might have forgot some other issue.

Not if "free worldwide" only refers to purchase price. Availability is a
different issue. When (if) the 40-bit law is no longer a factor the price
will be the same as it is now.

>Sorry world, you have to wait.

For now.

>Boy boy do I agree with you on this. As long as people keep spending
>money on beta test units and keep debugging for WHEELIE BILLY, the market
>will always exist. It seems to me that writing buggy program is good
business so
>that the sellers can say "Oh, you'll have to wait or upgrade the next
>version/release" I do use MS products, but not by choice.

I use Microsoft products, too, and by choice. The number of people using Microsoft products seems to indicate that it works pretty well, in spite of any bugs. Windows has gotten better with every release, as well it should have. It now has features that weren't even possible in Win3.1. Win95 and Win98 support for devices is also very extensive. I'm 52 years old, and I don't even have time left to READ 5,000,000 lines of code, much less try to write a competitor to Windows. Sure, it still has bugs in it. Some of them will never be found. Not because no one is looking for them, but because there is just too much ground to cover. Does that mean it should be scrapped? I don't think so, it still works well enough to suit me. It must work well enough to suit others, too. I don't see a mass movement to any other operating system, DOS or windows based. The future will tell.

Subject: Re: AES and patent rights
Date: 2 Oct 1998 09:19:28 GMT
From: Casper.Dik@Holland.Sun.Com (Casper H.S. Dik - Network Security Engineer)
Message-ID: <casper.907319695@nl-usenet.sun.com>
References: <36127F94.5955@ssd.bna.boeing.com>
Newsgroups: sci.crypt
Lines: 25

[[PLEASE DON'T SEND ME EMAIL COPIES OF POSTINGS]]

"Joseph K. Nilaad" <jknilaad@ssd.bna.boeing.com> writes:

>Olympic? Hmm. Why did I see Magic Johnson, Michael Jordan, Graff, etc. in
>Bacelona, Spain? I thought the whole idea of Olympic atheletes is to bring
>out the *best* atheletes in each field from each country. Thanks for bringing
>Olympic scenario up, who is collecting the money? Certainly not the atheletes.
>I have not yet seen any free admission in Olympic.

The Olympics started out as a venture for amateurs only.

While this has been made to sound admirable, it really was only a ploy
to keep the "underclass" who couldn't possibly afford to compete for free
away. Women weren't allowed to compete either.

And it had nothing to do with the classical olympics; those were very much
for profit.

Casper
--
Expressed in this posting are my opinions. They are in no way related
to opinions held by my employer, Sun Microsystems.
Statements on Sun products included here are not gospel and may
be fiction rather than truth.

Subject: Re: AES and patent rights
Date: 30 Sep 1998 23:44:28 GMT
From: lamontg@bite.me.spammers
Message-ID: <6uufos$15v0$1@nntp6.u.washington.edu>
References: <3610a518.969320@news.erols.com>
Newsgroups: sci.crypt
Lines: 74

ritter@io.com (Terry Ritter) writes:
>I disagree. This idea of working for prestige demeans the creative
>element and, indeed, work itself:
>
>* Perhaps writers should write for the privilege of being published.

they not infrequently do.

>* Perhaps actors should act for the privilege of being seen.

they not infrequently do.

>* Perhaps we all should work for praise alone.

sounds good, i don't think it's a bad idea at all for everyone to do some
work for no tangible compensation. of course you probably meant that
nobody should get paid for *anything*, which is just a bad debating
tactic.

>Anyone deserves prestige for the quality of their work. But they also
>deserve to be compensated for *doing* the work.

no. if i go out and do something which i consider to be work and everyone

else considers to be worthless then i certainly don't deserve to be compensated for doing the work. and if i do work without entering into a contractual agreement with someone then they should not be obligated to pay me. i can also freely elect to do work for no compensation or for substantially less compensation than i would otherwise obtain. you seem to be arguing that volunteerism is somehow morally wrong.

>Here the work is
>design, and that work deserves compensation. But AES rewards fall to
>manufacturing, who get a design free. So even though the whole point
>of this will be to enable a profit-based industry of ciphering
>hardware and software, there is no profit for the designers.

that's right. it also enables non-profit-based industry of ciphering software -- e.g. PGP.

>This
>demeans cipher designers in general, and (what a surprise!) acts to
>prevent a profit-based industry of cipher design.

the AES contest is, unfortunately, not the totality of the cipher design industry.

i suggest that if you really have something better, that you publish your methods of breaking the AES candidates and sell your improved algorithm to corporations in the private sector who are willing to pay you for your services.

however, if you're interested in money i strongly suggest that you go into some other industry -- given the availability of algorithms like Blowfish, Twofish, TEA, 3DES, Serpent, Rijndael, SAFER and CAST-128 there isn't a whole lot of need for stronger for-profit crypto algorithm design unless you can break all of those. i suggest getting paid for doing implimentations and doing cipher design and cryptanalysis in your off-time.

oh, do you feel that NIST should have _created_ a market for for-profit ciphers? isn't that creating a demand when there isn't one, and isn't that an interference in the free market?

>>You chose not to accept
>>their offer. Others did choose to accept their offer. This is an
>>example of free trade. The fact that their offer of payment is in
>>intangibles doesn't change that.
>
>I believe it does.

but you still haven't explained how.

--
Lamont Granquist (lamontg@u.washington.edu)
looking for unix administration / security work

Subject: Re: AES and patent rights
Date: Tue, 29 Sep 1998 15:52:20 GMT
From: schneier@counterpane.com (Bruce Schneier)
Message-ID: <361101b6.5807814@news.visi.com>
References: <36108e34.28361720@news.io.com>
Newsgroups: sci.crypt
Lines: 29

On Tue, 29 Sep 1998 07:38:05 GMT, ritter@io.com (Terry Ritter) wrote:
>On Mon, 28 Sep 1998 13:25:01 GMT, in <360f8dcf.3312669@news.visi.com>,
>in sci.crypt schneier@counterpane.com (Bruce Schneier) wrote:
>
>>[...]
>>To me it looks like we were both given the same decision to make, and
>>you chose one path and I chose the other. You believed that your
>>patent rights were worth more than NIST was willing to give you for
>>them.
>
>This is sort of a strange comment, isn't it? It might even be the
>basis for a sort of occupational joke, where a mathematician gets
>"paid" with zero dollars and goes away satisfied! Ha ha, very funny!
>
>Had AES offered even token real compensation for these rights, you
>might have a point. They did not.

As funny as it may seem, that is exactly the deal that all of the AES submitters accepted. And it is the deal that you didn't accept. And I still believe that you and I both were given the same decision to make. And I still believe that we chose to respond differently. I don't think I have a "point," other than you chose not to submit and I chose to submit.

Bruce
****
Bruce Schneier, President, Counterpane Systems Phone: 612-823-1098
101 E Minnehaha Parkway, Minneapolis, MN 55419 Fax: 612-823-1590
Free crypto newsletter. See: http://www.counterpane.com

Subject: Re: AES and patent rights
Date: Tue, 29 Sep 1998 21:25:52 +0200

From: <tbb03ar@mail.lrz-muenchen.de>
Message-ID: <Pine.GSO.4.03.9809292117190.29627-100000@sun5>
References: <36108e34.28361720@news.io.com>
Newsgroups: sci.crypt
Lines: 41

On Tue, 29 Sep 1998, Terry Ritter wrote:

>
> On Mon, 28 Sep 1998 13:25:01 GMT, in <360f8dcf.3312669@news.visi.com>,
> in sci.crypt schneier@counterpane.com (Bruce Schneier) wrote:
>
> >[...]
> >To me it looks like we were both given the same decision to make, and
> >you chose one path and I chose the other. You believed that your
> >patent rights were worth more than NIST was willing to give you for
> >them.
>
> This is sort of a strange comment, isn't it? It might even be the
> basis for a sort of occupational joke, where a mathematician gets
> "paid" with zero dollars and goes away satisfied! Ha ha, very funny!
>
> Had AES offered even token real compensation for these rights, you
> might have a point. They did not.
>

AES wouldn't be worth anything if it would be patented: Nobody is willing
to pay for an algorithm if there are lots of others in the public domain.

To get a standard it was neccessary to find free programs.

But I don't see that there is any problem: You didn't make money with DES
and you won't make money with the next standard. NIST didn't need you to
get DES and they don't need you to get AES :)
Others think it is worth developing algorithms only to be come the person
or group that developed AES.

BTW: Do you think the development of GNU C is unfair against Borland and
Microsoft?

Andreas Enterrottacher

enterrottacher@t-online.de
enterrottacher@lrz.tu-muenchen.de

Subject: Re: AES and patent rights
Date: Tue, 29 Sep 1998 20:22:24 GMT
From: ritter@io.com (Terry Ritter)
Message-ID: <3611414c.1821978@news.io.com>
References: <Pine.GSO.4.03.9809292117190.29627-100000@sun5>
Newsgroups: sci.crypt
Lines: 43

On Tue, 29 Sep 1998 21:25:52 +0200, in
<Pine.GSO.4.03.9809292117190.29627-100000@sun5>, in sci.crypt
<tbb03ar@mail.lrz-muenchen.de> wrote:

>[...]
>AES wouldn't be worth anything if it would be patented: Nobody is willing
>to pay for an algorithm if there are lots of others in the public domain.

RSA.

RC4 (in the sense that it was easy to export).

(Both of which are not free.)

>To get a standard it was neccessary to find free programs.

First, AES is a cipher; a major component, assuredly, but still only
one component of a complete system. It is not a "program."

And while there may be some "free" programs which use AES, we can be
sure that commercial software firms will compensate their programmers
by charging for the software. Programmers thus will be compensated --
and justly so -- for the time they spend; but cipher designers will
not be compensated for the vastly greater time *they* spend. And
though I do wear both hats, I still find this irritating, since it is
a direct result of government action.

>[...]
>BTW: Do you think the development of GNU C is unfair against Borland and
>Microsoft?

I guess that would first depend upon whether the government was
supporting GNU C, and next whether the government would be
recommending GNU C and even requiring it for their own use.

Terry Ritter ritter@io.com http://www.io.com/~ritter/
Crypto Glossary 1998-08-27: http://www.io.com/~ritter/GLOSSARY.HTM

Subject: Re: AES and patent rights
Date: Wed, 30 Sep 1998 09:04:05 +0200
From: <tbb03ar@mail.lrz-muenchen.de>
Message-ID: <Pine.GSO.4.03.9809300841410.15029-100000@sun5>
References: <3611414c.1821978@news.io.com>
Newsgroups: sci.crypt
Lines: 41

On Tue, 29 Sep 1998, Terry Ritter wrote:

>
> On Tue, 29 Sep 1998 21:25:52 +0200, in
> <Pine.GSO.4.03.9809292117190.29627-100000@sun5>, in sci.crypt
> <tbb03ar@mail.lrz-muenchen.de> wrote:
>
> >[...]
> >AES wouldn't be worth anything if it would be patented: Nobody is willing
> >to pay for an algorithm if there are lots of others in the public domain.
>
> RSA.
>
> RC4 (in the sense that it was easy to export).
>
> (Both of which are not free.)
>
>

And since DH has become free RSA isn't used any more in PGP. As well
others don't use RSA because DH has become free.

RC4 is de facto free: Everybody does a small change and uses the resulting
cipher in free programs.

> >To get a standard it was neccessary to find free programs.
>
> First, AES is a cipher; a major component, assuredly, but still only
> one component of a complete system. It is not a "program."

Sorry for the mistake. I should have written 'it was neccessary to find
free ciphers'.

> ...

Andreas Enterrottacher

enterrottacher@lrz.tu-muenchen.de
enterrottacher@t-online.de

Subject: Re: AES and patent rights
Date: Wed, 30 Sep 1998 15:46:33 GMT
From: jsavard@tenMAPSONeerf.edmonton.ab.ca (John Savard)
Message-ID: <36124f26.2015479@news.prosurfr.com>
References: <Pine.GSO.4.03.9809300841410.15029-100000@sun5>
Newsgroups: sci.crypt
Lines: 37

<tbb03ar@mail.lrz-muenchen.de> wrote, in part:

>RC4 is de facto free: Everybody does a small change and uses the resulting
>cipher in free programs.

The point is, though, that some people do pay for RC4, because 40-bit
RC4 gets them through the export approval process quickly.

Unlike RSA - which has technical merit behind it, being more
convenient to use than Diffie-Hellman for many purposes - RC4 is the
beneficiary of a government monopoly; the people paying for 40-bit
RC4 would have been happy to use DES instead were it not for certain
laws.

Ironically, RC4 was never patented, but was protected instead as a
trade secret. It used to be that a trade secret, once it ceased to be
secret, had no legal protection at all (although there is certainly
legal recourse to people who obtain access to a trade secret by
trespass or violation of contract), but now the area is quite
confusing and complicated; the state of California, for example, has
extended trade secret laws IIRC. (However, my understanding is that
the free programs usually _don't_ make a small change to the
algorithm, they merely avoid using the "RC4" trademark.)

Also, the alleged RC4 algorithm has a considerable similarity to Terry
Ritter's own patented Dynamic Substitution algorithm. Apparently,
though, it misses infringing because the Dynamic Substitution patent's
claims refer only to the use of the construction as a new kind of

combiner, for the purpose of applying a stream cipher keystream directly to plaintext (including intermediate forms of plaintext on their way to becoming final ciphertext). Applying one stream output to another stream output prior to its application to plaintext was not claimed, either by simple omission, or because that case was too similar to the cited prior art of the MacLaren-Marsaglia PRNG.

John Savard
http://members.xoom.com/quadibloc/index.html

Subject: Re: AES and patent rights
Date: Mon, 28 Sep 1998 13:28:17 GMT
From: schneier@counterpane.com (Bruce Schneier)
Message-ID: <360f8e3c.3422058@news.visi.com>
References: <360f0886.6675979@news.io.com>
Newsgroups: sci.crypt
Lines: 35

On Mon, 28 Sep 1998 03:57:06 GMT, ritter@io.com (Terry Ritter) wrote:
>Academic works generally are *required* to acknowledge the sources of
>*ideas*, and this is often done even for "private communications" such
>as personal letters and even private discussions. These are certainly
>far more questionable than any published works.
>
>Web pages and Usenet articles are published worldwide in form fixed as
>of a specific date, should have the author's name and a title, and
>carry both a legal copyright and ethical pride-of-authorship. This is
>indeed "publication" for academic purposes. Electronic publication
>can establish legal and moral priority in a field, and is disregarded
>only by those who wish to be known as academic thieves.
>
>Again, my stuff is available free on my pages. Any alleged scientist
>in cryptography who hasn't kept up with it has nobody to blame but
>their own lazy self.

Probably, but it takes a lot of work to keep with the Internet, and
some might argue that it is impossible. There is just so much out
there, and such a high percentage of it is trash, that it just isn't
cost effective to wade through it all. For everyone like you, who is
doing serious work, there are dozens of yahoos who aren't. And from
first glance, you can't tell the difference.

I believe this is going to be true not just for cryptography, but for
everything else. In a world where everyone is a publisher, editors
become more vital. Those who self publish will just find themselves
more and more marginalized, as anyone who has searched for information
on Alta Vista can see.

Bruce
****
Bruce Schneier, President, Counterpane Systems Phone: 612-823-1098
101 E Minnehaha Parkway, Minneapolis, MN 55419 Fax: 612-823-1590
Free crypto newsletter. See: http://www.counterpane.com

Subject: Re: AES and patent rights
Date: Mon, 28 Sep 1998 15:26:53 GMT
From: nospam@pd.jaring.my (Lincoln Yeoh)
Message-ID: <360fa4d6.7266602@nntp.jaring.my>
References: <360f0886.6675979@news.io.com>
Newsgroups: sci.crypt
Lines: 43

On Mon, 28 Sep 1998 03:57:06 GMT, ritter@io.com (Terry Ritter) wrote:

>Clearly, if someone else used my work in their submission, I would
>"participate" in AES without loss to me. My patents would still apply
>even if that design was selected.
>
>But I could not *submit* *my* *own* *work* without loss of rights.
>
>This means that a government process -- one that should apply to me
>just the same as you -- would cost me *more* than it cost you. This
>is just not equal protection under the law.
>

It is true: you can't give and keep at the same time. I don't see how equal
protection under the law comes into this.

I see the AES as a donation of your work if you are selected. If you
aren't selected you get to keep it. No one is forcing you to give.

You don't have to donate your blood (sweat and tears) if you don't want to.
Perhaps you're anaemic and it'll cost you more, in which case don't donate.

You won't get people telling you how grateful they are for your gift and
sacrifice. My heart bleeds for you, not. :).

So what if Bruce has already given his blood to the blood banks, and thus
it "costs him less"(?).

If you think his blood is inferior and not suitable, do tell us why and

how.

Don't go to the Crypto Donation Drive, if you don't want to give.

Link.

p.s. Remember too- if your blood ain't suitable, they don't take it from you.
****
Reply to: @Spam to
lyeoh at @people@uu.net
pop.jaring.my @
****

Subject: Re: AES and patent rights
Date: 28 Sep 1998 16:36:49 GMT
From: lamontg@bite.me.spammers
Message-ID: <6uodv1$123u$1@nntp6.u.washington.edu>
References: <360f0886.6675979@news.io.com>
Newsgroups: sci.crypt
Lines: 32

ritter@io.com (Terry Ritter) writes:
[...]
>Well, this is progress! Now we're about halfway there:
>
>Clearly, if someone else used my work in their submission, I would
>"participate" in AES without loss to me. My patents would still apply
>even if that design was selected.

That isn't at all obvious. At any rate, NIST will probably not select an algorithm which uses prior work that might be patented -- at least they should disqualify algorithms where there are questions raised about this. In the case of twofish, Bruce seems to have a pretty good idea of the "intellectual history" of his cipher so he should be able to make a fairly good case for it containing un-patentable algorithms -- other algorithms should do the same.

[...]
>Again, my stuff is available free on my pages. Any alleged scientist
>in cryptography who hasn't kept up with it has nobody to blame but
>their own lazy self.

Yeah, so now every cryptographer our there is expected to do frequent and comprehensive web searches and stay abrest of Usenet newsgroups or else they are "lazy." *You* are the one that is lazy. Writing an article and sticking it up on a website is something that any college student can do. Doing the work to make it suitable for publication in a journal takes substantially longer. That standard of quality is why "lazy" cryptographers read journals rather than browse random web pages.

--
Lamont Granquist (lamontg@u.washington.edu)
looking for unix administration / security work

Subject: Re: AES and patent rights
Date: Mon, 28 Sep 1998 16:03:58 -0600
From: jgfunj@EnqvbSerrGrknf.pbz (W T Shaw)
Message-ID: <jgfunj-2809981603590001@dialup154.itexas.net>
References: <6uodv1$123u$1@nntp6.u.washington.edu>
Newsgroups: sci.crypt
Lines: 72

In article <6uodv1$123u$1@nntp6.u.washington.edu>,
lamontg@bite.me.spammers wrote:
>
> Yeah, so now every cryptographer our there is expected to do frequent and
> comprehensive web searches and stay abrest of Usenet newsgroups or else
> they are "lazy." *You* are the one that is lazy. Writing an article and
> sticking it up on a website is something that any college student can do.
> Doing the work to make it suitable for publication in a journal takes
> substantially longer. That standard of quality is why "lazy" cryptographers
> read journals rather than browse random web pages.
>
It all depends on whether you actually want to keep up with a fast changing field or not; this affects everything, not cryptography alone. I would suggest that you discard email as well since, according to the same reasoning, nothing not carefully written out and properly mailed would constitute laziness. I guess doing away with computers could be also justified that way.

Consider the importance of form vs. substance. Learning the approved form is not essential with grasping substance. The internet facilitates communication, and works to shortcircuit obstacles. To get to new information, the removal of hinderances should be everyones priority. You certailnly object when government attempts to restrict spread of your information.

The formal requirement to jump throught x number of hoops before anyone will listen is simply a mistatement of the facts that people much can be accomplished in nontraditional ways.

The internet is causing a shift, from established entities as centers, who assumed that they were gatekeepers, the lord protectors of something they could define and redefine as they saw fit, to the democratic voices of all who what to be involved. Those in power centers who accomidate and use this new input face brighter futures than those who cannot adapt.

The most obvious change is the fruitbasket turnover of the way the news media works, or has been incapable of maintaining its old identity lately. It will get sorted out somehow, but none can predict how.

Requiring that everything pipe through approved channels is simply the same the old governmental position that they can force such, control the channels, and therefore manage communications whether anyone likes it or not. The biggest single item is that it is increasingly difficult to mask dissent, something our founding fathers would cheer about had they seen it coming. The very fall of the Soviet Union is keyed to no longer being able to run all communications through government turnstyles.

Requirements for formalism can mean just as much, it just depends who the masters are, which in too many cases can mean adopting self-serving arbitrariness of tenured professors, who do not give a good name to authoritarian.

Thankfully, most of them in cryptography are sincere, sympathetic, and reasonable. But, I've met a few prima donnas who would decry the internet as spoiling the value of their publish-or-perish world. The ones that understand are most likely to be fully into the electronic media as well.

All media should complement each other, not kick each other around so as to bury vital information. To demand otherwise is pro-censorship, something you surely would not be in favor of if given the choice. You cannot stop the world if things get too fast for you. Nor, can you expect everyone to run the same crinulated pattern you might have taken.
--

Are you tired, rundown, can't find a corner in the office to hide in?

Then, try Jimmy Carter's Little Pills, which are apt to cause you to want to get out your frustrations *constructively*, but might tend to make you fear rabbits.

Decrypt with ROT13 to get correct email address.

Subject: Re: AES and patent rights
Date: 30 Sep 1998 04:13:42 GMT
From: lamontg@bite.me.spammers
Message-ID: <6usb5mnic1@nntp6.u.washington.edu>
References: <6uodv1$123u$1@nntp6.u.washington.edu>
Newsgroups: sci.crypt
Lines: 58

Look, I have been on the Internet since 1989 and I've made exactly the same arguments that you have. The fact is, however, that the author of some work bears a responsibility to attempt to get the word out. I entirely appreciate the authoritarian argument, but some centralization is simply necessary.

I think, however, that this centralization can take many forms. Certainly anyone is free to put up their work on their private web pages. I think also that people should look to examples like the physics pre-print server at xxx.lanl.gov as being a model of how to communicate rapidly, informally and without as many "authoritarian" controls.

However, the existance of refereed print journals (or e-print journals) is still going to be necessary since they do enforce a very high standard. You simply will not get high quality refereeing and low signal to noise ratio without paying people to be gatekeepers of information. This then sets a standard which other sources of information can strive to achieve -- while they may distribute otherwise inaccessable information, which might be rejected by the "authoritarian" controls. I'm fine with this.

And I wouldn't arbitrarily throw out anything which wasn't published in a refereed journal, but I think the author should have some responsbility to attempt to publish it in semi-formal forums like the physics pre-print server and to attempt to get it into the refereed journals. Ideally they should also attempt to come to conferences and spread their ideas. Saying "I put it up on my web site 6 months ago, why haven't you read it?" is egotistical b.s. and it isn't publishing -- getting it into central repositories of information is publishing.

jgfunj@EnqvbSerrGrknf.pbz (W T Shaw) writes:
[...]
>The formal requirement to jump throught x number of hoops before anyone
>will listen is simply a mistatement of the facts that people much can be
>accomplished in nontraditional ways.
>
>The internet is causing a shift, from established entities as centers, who
>assumed that they were gatekeepers, the lord protectors of something they
>could define and redefine as they saw fit, to the democratic voices of all
>who what to be involved. Those in power centers who accomidate and use
>this new input face brighter futures than those who cannot adapt.
[...]

>Requiring that everything pipe through approved channels is simply the
>same the old governmental position that they can force such, control the
>channels, and therefore manage communications whether anyone likes it or
>not. The biggest single item is that it is increasingly difficult to
>mask dissent, something our founding fathers would cheer about had they
>seen it coming. The very fall of the Soviet Union is keyed to no longer
>being able to run all communications through government turnstyles.
[...]
>All media should complement each other, not kick each other around so as
>to bury vital information. To demand otherwise is pro-censorship,
>something you surely would not be in favor of if given the choice. You
>cannot stop the world if things get too fast for you. Nor, can you expect
>everyone to run the same crinulated pattern you might have taken.

--
Lamont Granquist (lamontg@u.washington.edu)
looking for unix administration / security work

Subject: Re: AES and patent rights
Date: Mon, 28 Sep 1998 00:53:27 -0600
From: jgfunj@EnqvbSerrGrknf.pbz (W T Shaw)
Message-ID: <jgfunj-2809980053280001@207.22.198.201>
References: <360ed738.1402804@news.visi.com>
Newsgroups: sci.crypt
Lines: 76

In article <360ed738.1402804@news.visi.com>, schneier@counterpane.com
(Bruce Schneier) wrote:

>
> Unfortunately, that's not true. (And it is unfortunate.) Publication
> does not mean self-publication on a website, it means publication in a
> workshop, conference, or journal.

Publication means, amongst other things, "3. Communication of information
to the public."

We are at a great transition where actual printed matter is fastly being
usurped by electronic media...it's in all the *papers*. Holding to the
old standard as the one true path is merely quaint.

>
> In any case, even if you don't want to publish in conferences or
> journals, put cryptanalysis papers on your website. As I said before,
> new ideas just for their own sake aren't very interesting. You need
> to show how the old ideas are insufficient. You need to break ciphers
> designed with the old ideas, and then show how your own ideas are
> better.
>
> Designs are dime a dozen, so it's hard to seperate the good ones from
> the silly ones. Good cryptanalysis is hard; it will force people to
> take notice of your work.

The big word generally preached as gospel for ages has that the world
would be your oyster if anyone could come up with a really good
algorithm.

Things have changed, it seems....the welcome wagon is no longer out.

I remember not too long ago when you dwelt on how difficult writing crypto
was. . I suppose that certain algorithms are harder to write than others.
Yet, that does not necessarily mean ones more difficult to do are better
than all others.

The harder it is to sort algorithms out, the more implicitly they must all
tend to be similiar in goodness, strenght, etc., to each other. To
criticise any particular one that you cannot break yourself, if noone else
has, as silly would be rather unscientific.

Something is to be said about analysis of weaker ciphers, something that I
do on a limited scale routinely, but wish I had more time to do. I enjoy
those pursuits as it can be most satisfying to get into the mind of the
one who contrived a devilishly difficult cipertext. The varied lessons
are all relevant, even from that level to sophistocated ones since logic
is common in all of these things. It is simply a matter of economy where
one places his time, something you cannot determine for anyone but
yourself.

To demand anyone must break an old algorithm to be noticed as a hurdle is
too much given the effort required to preform feats and the sparce nature
of such challenges. You might spend lots of effort trying to break one
that would surpass all others as well. It is rather tempting to try to
make algorithms that could be broken, just to increase the supply, and
therefore qualify more people into a higher realm.

I would not discount your formula; it should, however, not be the only
formula. Making and breaking algorithms is surely not always highly
correlated.

A word about Terry Ritter, he has experience the *ugly* end of the
government crypto stick in the past in ways that you obviously have not.
It is his story to tell if he wishes. I understand why he is not willing
to let something similiar happen again.

--

Are you tired, rundown, can't find a corner in the office to hide in?

Then, try Jimmy Carter's Little Pills, which are apt to cause you to
want to get out your frustrations *constructively*, but might tend
to make you fear rabbits.

Decrypt with ROT13 to get correct email address.

Subject: Re: AES and patent rights
Date: Mon, 28 Sep 1998 13:16:27 GMT
From: schneier@counterpane.com (Bruce Schneier)
Message-ID: <360f8959.2170729@news.visi.com>
References: <jgfunj-2809980053280001@207.22.198.201>
Newsgroups: sci.crypt
Lines: 140

On Mon, 28 Sep 1998 00:53:27 -0600, jgfunj@EnqvbSerrGrknf.pbz (W T
Shaw) wrote:

>In article <360ed738.1402804@news.visi.com>, schneier@counterpane.com
>(Bruce Schneier) wrote:
>
>>
>> Unfortunately, that's not true. (And it is unfortunate.) Publication
>> does not mean self-publication on a website, it means publication in a
>> workshop, conference, or journal.
>
>Publication means, amongst other things, "3. Communication of information
>to the public."
>
>We are at a great transition where actual printed matter is fastly being
>usurped by electronic media...it's in all the *papers*. Holding to the
>old standard as the one true path is merely quaint.

The difference is not "dead trees versus electrons." The difference
is "recognised publication versus self publication." If random person
X puts some kind of crypto design paper on his website, almost no one
in the community will read it. Yes, it is available to them. Yes,
they have the technical ability to read it. But they have no idea if
it is a waste of time to read or not.

Anyone can put their ideas up on the web; it's the ultimate vanity
press. But there is just too much out there; the community needs some
way to determine if a particular something is worth reading. The way
the community uses is publication. This way is inperfect, and not
without flaws, but it's pretty good.

Things that aren't published just aren't recognised by the community.

When I submitted my second related-key cryptanalysis to a conference
for the first time, it contained a cryptanalysis of an algorithm that
appeared on the web. It illustrated the attack nicely, and I thought
it was a good addition. One reviewer commented, in his anonymous
review, that ciphers posted on the web are not worth breaking. We had
to take that section out of the paper before publication.

Now you can argue whether or not that is a good thing, but that's the
way the world works.

>> In any case, even if you don't want to publish in conferences or
>> journals, put cryptanalysis papers on your website. As I said before,
>> new ideas just for their own sake aren't very interesting. You need
>> to show how the old ideas are insufficient. You need to break ciphers
>> designed with the old ideas, and then show how your own ideas are
>> better.
>>
>> Designs are dime a dozen, so it's hard to seperate the good ones from
>> the silly ones. Good cryptanalysis is hard; it will force people to
>> take notice of your work.
>
>The big word generally preached as gospel for ages has that the world
>would be your oyster if anyone could come up with a really good
>algorithm.
>
>Things have changed, it seems....the welcome wagon is no longer out.

Indeed it isn't. Algorithms are easy, and it is hard to figure out
whether or not something is really good. I get about two algorithms a
week in letters and email. I don't have the time or patience to wade
through every one of them looking for the few good ideas. Designers
are expected to do their own analysis work.

>I remember not too long ago when you dwelt on how difficult writing crypto
>was. . I suppose that certain algorithms are harder to write than others.
>Yet, that does not necessarily mean ones more difficult to do are better
>than all others.

It's difficult to design and analyze a new algorithm; just creating
one is easy.

>The harder it is to sort algorithms out, the more implicitly they must all

>tend to be similiar in goodness, strenght, etc., to each other. To
>criticise any particular one that you cannot break yourself, if noone else
>has, as silly would be rather unscientific.

I don't think I am doing that. Remember, just because no one has
broken a cipher does not mean that it is secure. If everyone has
TRIED to break the cipher and no one has broken it, that's a much
better indication.

The problem is that random ciphers posted on the Internet are just not
looked at, so no one knows if they are good or not. They are not
looked at because if we break them, we can't even publish our work.
That's not much incentive, when there are fifteen AES candidates out
there whose breaks can be published.

Again, it's imperfect, but there really isn't any good alternative.
There are only so many cryptanalysis hours in a day.

>Something is to be said about analysis of weaker ciphers, something that I
>do on a limited scale routinely, but wish I had more time to do. I enjoy
>those pursuits as it can be most satisfying to get into the mind of the
>one who contrived a devilishly difficult cipertext. The varied lessons
>are all relevant, even from that level to sophistocated ones since logic
>is common in all of these things. It is simply a matter of economy where
>one places his time, something you cannot determine for anyone but
>yourself.

Agreed. We break weaker things all the time: the weak AES candidates,
the random stupid cellphone algorithms. If I am going to break some
amateur design, I prefer that it be something in a well-known product.
Then, at least, I can make a little PR noise.

>To demand anyone must break an old algorithm to be noticed as a hurdle is
>too much given the effort required to preform feats and the sparce nature
>of such challenges. You might spend lots of effort trying to break one
>that would surpass all others as well. It is rather tempting to try to
>make algorithms that could be broken, just to increase the supply, and
>therefore qualify more people into a higher realm.

I want it to be a difficult hurdle. Anyone can create a cipher that
he himself cannot break. Before I look at a cipher that someone
created and cannot break, I want some indication that his lack of
ability to break the cipher means something. And honestly, there is
still a lot of low hanging fruit out there; published ciphers that are
very breakable.

>I would not discount your formula; it should, however, not be the only
>formula. Making and breaking algorithms is surely not always highly
>correlated.

Probably not, but it is the only reasonable formula I can think of.

>A word about Terry Ritter, he has experience the *ugly* end of the
>government crypto stick in the past in ways that you obviously have not.
>It is his story to tell if he wishes. I understand why he is not willing
>to let something similiar happen again.

This is not meant to be a person attack in any way. I'm sorry if he
feels that way.

Bruce
****
Bruce Schneier, President, Counterpane Systems Phone: 612-823-1098
101 E Minnehaha Parkway, Minneapolis, MN 55419 Fax: 612-823-1590
Free crypto newsletter. See: http://www.counterpane.com

Subject: Re: AES and patent rights
Date: Tue, 29 Sep 1998 05:22:31 GMT
From: "Douglas A. Gwyn" <DAGwyn@null.net>
Message-ID: <36106E74.D61C1C92@null.net>
References: <360f8959.2170729@news.visi.com>
Newsgroups: sci.crypt
Lines: 13

Bruce Schneier wrote:
> When I submitted my second related-key cryptanalysis to a conference
> for the first time, it contained a cryptanalysis of an algorithm that
> appeared on the web. It illustrated the attack nicely, and I thought
> it was a good addition. One reviewer commented, in his anonymous
> review, that ciphers posted on the web are not worth breaking. We had
> to take that section out of the paper before publication.

The reviewer clearly exhibited some snobbery there;
most of the truly classic papers on cryptanalysis
used as examples systems that had never been published.
If it makes a good example, it is pedagogically useful
to have it in your article, regardless of its origin.

Subject: Re: AES and patent rights
Date: Tue, 29 Sep 1998 15:53:17 GMT
From: schneier@counterpane.com (Bruce Schneier)
Message-ID: <3611024a.5955845@news.visi.com>

References: <36106E74.D61C1C92@null.net>
Newsgroups: sci.crypt
Lines: 26

On Tue, 29 Sep 1998 05:22:31 GMT, "Douglas A. Gwyn" <DAGwyn@null.net>
wrote:

>Bruce Schneier wrote:
>> When I submitted my second related-key cryptanalysis to a conference
>> for the first time, it contained a cryptanalysis of an algorithm that
>> appeared on the web. It illustrated the attack nicely, and I thought
>> it was a good addition. One reviewer commented, in his anonymous
>> review, that ciphers posted on the web are not worth breaking. We had
>> to take that section out of the paper before publication.
>
>The reviewer clearly exhibited some snobbery there;
>most of the truly classic papers on cryptanalysis
>used as examples systems that had never been published.
>If it makes a good example, it is pedagogically useful
>to have it in your article, regardless of its origin.

I agree with you, more or less. But this is the reality of academic
cryptography. I cannot change that, even if I sit on program
committees and argue the point.

Bruce
****
Bruce Schneier, President, Counterpane Systems Phone: 612-823-1098
101 E Minnehaha Parkway, Minneapolis, MN 55419 Fax: 612-823-1590
Free crypto newsletter. See: http://www.counterpane.com

Subject: Re: AES and patent rights
Date: Tue, 29 Sep 1998 13:13:52 GMT
From: "Joseph K. Nilaad" <jknilaad@ssd.bna.boeing.com>
Message-ID: <3610DD10.7753@ssd.bna.boeing.com>
References: <360f8959.2170729@news.visi.com>
Newsgroups: sci.crypt
Lines: 84

Bruce Schneier wrote:
>
> On Mon, 28 Sep 1998 00:53:27 -0600, jgfunj@EnqvbSerrGrknf.pbz (W T
> Shaw) wrote:
>
> >
> >Publication means, amongst other things, "3. Communication of information
> >to the public."
> >
> >We are at a great transition where actual printed matter is fastly being
> >usurped by electronic media...it's in all the *papers*. Holding to the
> >old standard as the one true path is merely quaint.
>
> The difference is not "dead trees versus electrons." The difference
> is "recognised publication versus self publication." If random person
> X puts some kind of crypto design paper on his website, almost no one
> in the community will read it. Yes, it is available to them. Yes,
> they have the technical ability to read it. But they have no idea if
> it is a waste of time to read or not.
If it is good for the goose, it's also good for the gander. Do you
think *everybody* will read all the works published *on paper* by NIST?
Beside paper copies may not be free and take too long to be delivered.

>
> Anyone can put their ideas up on the web; it's the ultimate vanity
> press. But there is just too much out there; the community needs some
> way to determine if a particular something is worth reading. The way
> the community uses is publication. This way is inperfect, and not
> without flaws, but it's pretty good.
How can you quantify which publication is worth reading either it is on
the web or hard copy? Whether it is on the web or hard copy, there
exists some merits to it.

>
> Things that aren't published just aren't recognised by the community.
Are you referring to government sponsored community? Why must one's
work
be endorsed by an organization and putting on paper in order to be
recognized by the "community". To me, recognition is majority of public
acceptance. For example, Windows was not endorsed by any organization;
however, the majority of public accept it.

>
> Indeed it isn't. Algorithms are easy, and it is hard to figure out
> whether or not something is really good. I get about two algorithms a
I have to agree with you here.

>
> >I remember not too long ago when you dwelt on how difficult writing crypto
> >was. . I suppose that certain algorithms are harder to write than others.
> >Yet, that does not necessarily mean ones more difficult to do are better
> >than all others.
>
> It's difficult to design and analyze a new algorithm; just creating

> one is easy.
Is this anamoly? Isn't that normally before you create something you must have some procedures in mind how about doing it first? Maybe you should talk to him about his algorithm, it is simple and it doesn't take that long to figure out that his stuff is pretty good.

>
> I don't think I am doing that. Remember, just because no one has
> broken a cipher does not mean that it is secure. If everyone has
> TRIED to break the cipher and no one has broken it, that's a much
> better indication.
How can you get *everyone* to TRIED to break a given cipher? if "no one has broken a cipher", it is secure! Think about it.

Anyway this thread is getting to long. Since you're involved with AES, I
have a few questions to you:

1. When NIST said that AES is to be "free world wide", will NIST make exception to export law which limit to 40 bits?

2. How will NIST deal with key escrow stuffs or will there be any key escrow issue?

3. Where is available site to find out about AES candidates?

Joe Nilaad
Nature is simple and beautiful...

Subject: Re: AES and patent rights
Date: 29 Sep 1998 10:47:06 -0500
From: juola@mathcs.duq.edu (Patrick Juola)
Message-ID: <6uqvdq$i2$1@quine.mathcs.duq.edu>
References: <3610DD10.7753@ssd.bna.boeing.com>
Newsgroups: sci.crypt
Lines: 61

In article <3610DD10.7753@ssd.bna.boeing.com>,
Joseph K. Nilaad <jknilaad@ssd.bna.boeing.com> wrote:
>Bruce Schneier wrote:
>>
>> On Mon, 28 Sep 1998 00:53:27 -0600, jgfunj@EnqvbSerrGrknf.pbz (W T
>> Shaw) wrote:
>>
>> >
>> >Publication means, amongst other things, "3. Communication of information
>> >to the public."
>> >
>> >We are at a great transition where actual printed matter is fastly being
>> >usurped by electronic media...it's in all the *papers*. Holding to the
>> >old standard as the one true path is merely quaint.
>>
>> The difference is not "dead trees versus electrons." The difference
>> is "recognised publication versus self publication." If random person
>> X puts some kind of crypto design paper on his website, almost no one
>> in the community will read it. Yes, it is available to them. Yes,
>> they have the technical ability to read it. But they have no idea if
>> it is a waste of time to read or not.
>If it is good for the goose, it's also good for the gander. Do you
>think *everybody* will read all the works published *on paper* by NIST?
>Beside paper copies may not be free and take too long to be delivered.

Well, no. Because NIST has something that most people don't -- to wit, credibility. The reason that NIST has credibility is because of something else that NIST has that most people haven't -- technical expertise and knowledge of the subject.

So the important thing about papers published by NIST is not that they are on paper, but that they're published under the aegis and auspices of someone who knows what they're talking about. The current gold standard for establishing credibility (at least in the sciences) is "peer review" -- to wit, having a bunch of established experts look at any particular contribution and figure out if it's worth reading. If they decide that it is, then it will be published (and read, and examined in detail), irrespective of the method of publication -- I've seen several electronic-only peer reviewed journals. But because peer-review is a lengthy and expensive process, most people want the "worthwhile" contributions to be put on a medium a little bit less transient than a web page -- and most organizations want to be able to charge money to cover the costs of the review.

The problem with "free information" is that most of the time it's worth what you pay for it. The advantage of information that you pay for is that it's more likely to be worth the cost.

>>
>> Things that aren't published just aren't recognised by the community.
>Are you referring to government sponsored community? Why must one's
>work
>be endorsed by an organization and putting on paper in order to be
>recognized by the "community". To me, recognition is majority of public
>acceptance. For example, Windows was not endorsed by any organization;

>however, the majority of public accept it.

Evidently Microsoft doesn't count as an organization? This is odd....

-kitten

Subject: Re: AES and patent rights
Date: Tue, 29 Sep 1998 16:00:27 GMT
From: schneier@counterpane.com (Bruce Schneier)
Message-ID: <36110282.6012001@news.visi.com>
References: <3610DD10.7753@ssd.bna.boeing.com>
Newsgroups: sci.crypt
Lines: 70

On Tue, 29 Sep 1998 13:13:52 GMT, "Joseph K. Nilaad"
<jknilaad@ssd.bna.boeing.com> wrote:
>Bruce Schneier wrote:
>> The difference is not "dead trees versus electrons." The difference
>> is "recognised publication versus self publication." If random person
>> X puts some kind of crypto design paper on his website, almost no one
>> in the community will read it. Yes, it is available to them. Yes,
>> they have the technical ability to read it. But they have no idea if
>> it is a waste of time to read or not.
>If it is good for the goose, it's also good for the gander. Do you
>think *everybody* will read all the works published *on paper* by NIST?
>Beside paper copies may not be free and take too long to be delivered.

Again, it's not the medium. If academic cryptographer A reads and
cryptanalyzes a method that appears at a conference, workshop, or as
an AES submission, he can publish his results. If he reads and
cryptanalyzes a method that appears on Usenet, all he can do is post
his results on Usenet. Now some people do this, but many people do
not. It's not a perfect filter--some of the AES submissions were
really lousy and some Usenet posts are good--but it's the filter that
most academics use.

>> Anyone can put their ideas up on the web; it's the ultimate vanity
>> press. But there is just too much out there; the community needs some
>> way to determine if a particular something is worth reading. The way
>> the community uses is publication. This way is inperfect, and not
>> without flaws, but it's pretty good.
>How can you quantify which publication is worth reading either it is on
>the web or hard copy? Whether it is on the web or hard copy, there
>exists some merits to it.

You can't. The premise is that it is impossible to read everything.
People have to filter in some way. Most academic cryptographers use
the publication process as a way to filter. Again, it is not a
perfect process.

>> Things that aren't published just aren't recognised by the community.
>Are you referring to government sponsored community? Why must one's work
>be endorsed by an organization and putting on paper in order to be
>recognized by the "community". To me, recognition is majority of public
>acceptance. For example, Windows was not endorsed by any organization;
>however, the majority of public accept it.

By "community" I mean research community.

>> It's difficult to design and analyze a new algorithm; just creating
>> one is easy.
>Is this anamoly? Isn't that normally before you create something you
>must have some procedures in mind how about doing it first? Maybe you
>should talk to him about his algorithm, it is simple and it doesn't take
>that long to figure out that his stuff is pretty good.

It's somewhat an anomoly, since functionality is orthogonal to
security. If someone invents a new compression algorithm, it's easy
to hand him the standard compression benchmarks and ask him to test
his algorithm against them. If someone invents a new factoring
algorithm, it's easy to tell him to factor the next RSA challenge
before he talks to you. (This got rid of many, many factoring
crackpots that would call RSADSI.) But if someone has a new
encryption algorithm, you can't just give him a battery of tests and
tell him to go away.

For me to look at an algorithm that someone sends me, it has to have
some serious advantage over the currently existing alternatives.

Bruce
****
Bruce Schneier, President, Counterpane Systems Phone: 612-823-1098
101 E Minnehaha Parkway, Minneapolis, MN 55419 Fax: 612-823-1590
Free crypto newsletter. See: http://www.counterpane.com

Subject: Re: AES and patent rights
Date: Wed, 30 Sep 1998 04:37:48 GMT
From: ritter@io.com (Terry Ritter)
Message-ID: <3611b512.31466865@news.io.com>
References: <36110282.6012001@news.visi.com>
Newsgroups: sci.crypt

Lines: 105

On Tue, 29 Sep 1998 16:00:27 GMT, in <36110282.6012001@news.visi.com>,
in sci.crypt schneier@counterpane.com (Bruce Schneier) wrote:

>
>Again, it's not the medium. If academic cryptographer A reads and
>cryptanalyzes a method that appears at a conference, workshop, or as
>an AES submission, he can publish his results. If he reads and
>cryptanalyzes a method that appears on Usenet, all he can do is post
>his results on Usenet.

The reality I see is that an academic can write -- and publish --
about anything that reveals new information. Various "papers" have
been written about toy ciphers that the author herself has simply
thought up, and which therefore also have no previous existence in the
academic literature.

As a matter of fact, now that I think of it, Biham *actually* *did*
write and publish an academic "paper" on *my* *own* "Ladder DES"
proposal, which was basically a Usenet thing. Indeed, as I recall,
you were somewhat involved in this, as you also have been on other
occasions. So, clearly, one *can* write about Usenet proposals.

In my particular case, many of my "methods that appear on Usenet" have
a basis in issued patents. Now, you may not *like* patents, but they
are *more* of an "archival publication" than any conference
proceeding. I would say that an expert is expected to know the state
of the art, and not just the state of the academic literature.

Yet we still have not seen my Dynamic Substitution -- a 1990 patent --
described in AC, have we?

>Now some people do this, but many people do
>not. It's not a perfect filter--some of the AES submissions were
>really lousy and some Usenet posts are good--but it's the filter that
>most academics use.

The whole process is incestuous. Everybody works on what everybody
"knows" is significant. The situation is ripe for some young Turks to
open up whole new classes of structure about which the older guys
simply have no clue.

>[...]
>The premise is that it is impossible to read everything.

It may be impossible for one person to read everything. But that does
not mean that each person cannot expand their horizons beyond the
current group reading assignment. And if someone finds something
interesting, they might recommend it to others -- even if it is not on
a bookshelf.

>People have to filter in some way. Most academic cryptographers use
>the publication process as a way to filter. Again, it is not a
>perfect process.

Peer-reviewed publication is certainly not perfect, and it is my
impression that it has gotten worse. You can blame the net if you
wish, but the real problem is that there is just more information.
This has meant increasing numbers of journals, and a general inability
of editors to perceive the import of new work, or detect gobbledygook.

The advantages of the "archival journal" system are less than they
were, and the disadvantage of a year or two publication delay is
increasingly significant to an author.

>But if someone has a new
>encryption algorithm, you can't just give him a battery of tests and
>tell him to go away.
>
>For me to look at an algorithm that someone sends me, it has to have
>some serious advantage over the currently existing alternatives.

I support this, but it is all too easy to think, "What I really want
is just like DES, with a few changes." And that is much too small a
box to fit new architectures into.

For example, a block cipher with huge blocks can:

* Avoid CBC chaining, thus allowing blocks to be ciphered
independently. This can translate to speed with parallel hardware,
and efficiency by ciphering packets even out of order.

* Support a "validation field" within the plaintext data block. By
changing the value in that field we have a homophonic block cipher.
This supports block-by-block data block validation without other
computation, which also can translate into speed.

Variable Size Block Ciphers also have advantages which are not

apparent in the context of a DES-style "cipher box."

My point is that seeing the "serious advantage" of new ciphers may
require some re-understanding of what a "cipher" is. But then we may
find that the new cipher is actually a better fit to the rest of the
system than the old DES box was.

Terry Ritter ritter@io.com http://www.io.com/~ritter/
Crypto Glossary 1998-08-27: http://www.io.com/~ritter/GLOSSARY.HTM

Subject: Re: AES and patent rights
Date: Wed, 30 Sep 1998 19:24:59 GMT
From: schneier@counterpane.com (Bruce Schneier)
Message-ID: <36128411.17019891@news.visi.com>
References: <3611b512.31466865@news.io.com>
Newsgroups: sci.crypt
Lines: 99

On Wed, 30 Sep 1998 04:37:48 GMT, ritter@io.com (Terry Ritter) wrote:
>As a matter of fact, now that I think of it, Biham *actually* *did*
>write and publish an academic "paper" on *my* *own* "Ladder DES"
>proposal, which was basically a Usenet thing. Indeed, as I recall,
>*you* were somewhat involved in this, as you also have been on other
>occasions. So, clearly, one *can* write about Usenet proposals.

Of course one can. Biham's paper is certainly a good example.

>In my particular case, many of my "methods that appear on Usenet" have
>a basis in issued patents. Now, you may not *like* patents, but they
>are *more* of an "archival publication" than any conference
>proceeding. I would say that an expert is expected to know the state
>of the art, and not just the state of the academic literature.

I've tried referencing patents, and while I get a better reception
than I do with Usenet posting, it's still dicey. There's a zeta
function cryptosystem that we analyzed for a client; we can publish
our work once the patent issues. We'll see if we can get that paper
accepted anywhere.

>Yet we still have not seen my Dynamic Substitution -- a 1990 patent --
>described in AC, have we?

Don't worry. I have all your patents. And I get an alert when a new
one is issued. (I assume others are in the pipeline.)

>The whole process is incestuous. Everybody works on what everybody
>"knows" is significant. The situation is ripe for some young Turks to
>open up whole new classes of structure about which the older guys
>simply have no clue.

I, for one, would love that. Do it. Rip through the field with some
devistating cryptanalysis that breaks things left and right. One nice
thing about cryptography is that if have have a new attack, it's hard
not to be noticed.

>Peer-reviewed publication is certainly not perfect, and it is my
>impression that it has gotten worse. You can blame the net if you
>wish, but the real problem is that there is just more information.
>This has meant increasing numbers of journals, and a general inability
>of editors to perceive the import of new work, or detect gobbledygook.

I agree that it has gotten worse, at least in cryptography. There are
far too many workshops and conferences that take pretty much anything
in the cryptography field.

>The advantages of the "archival journal" system are less than they
>were, and the disadvantage of a year or two publication delay is
>increasingly significant to an author.

Agreed as well. The Journal of Cryptology has something like a two
year delay, which is absurd. I know of many people who are not
bothering with the journal because of that. Workshops are better;
the deadline fo the March 1999 Fast Software Encrytption workshop is
in December.

>>But if someone has a new
>>encryption algorithm, you can't just give him a battery of tests and
>>tell him to go away.
>>
>>For me to look at an algorithm that someone sends me, it has to have
>>some serious advantage over the currently existing alternatives.
>
>I support this, but it is all too easy to think, "What I really want
>is just like DES, with a few changes." And that is much too small a
>box to fit new architectures into.

I agree, actually. This is why FSE still takes cipher designs, and
papers about new architectures (even though most of them are
medeocre).

>For example, a block cipher with huge blocks can:

>
>* Avoid CBC chaining, thus allowing blocks to be ciphered
>independently. This can translate to speed with parallel hardware,
>and efficiency by ciphering packets even out of order.
>
>* Support a "validation field" within the plaintext data block. By
>changing the value in that field we have a homophonic block cipher.
>This supports block-by-block data block validation without other
>computation, which also can translate into speed.
>
>Variable Size Block Ciphers also have advantages which are not
>apparent in the context of a DES-style "cipher box."
>
>My point is that seeing the "serious advantage" of new ciphers may
>require some re-understanding of what a "cipher" is. But then we may
>find that the new cipher is actually a better fit to the rest of the
>system than the old DES box was.

I agree with you 100%. Write it. Submit it to FSE. I am on the
committee. If it is a halfway decent paper, I will support it.

Bruce
****
Bruce Schneier, President, Counterpane Systems   Phone: 612-823-1098
101 E Minnehaha Parkway, Minneapolis, MN 55419   Fax: 612-823-1590
           Free crypto newsletter. See: http://www.counterpane.com

---

Subject: Re: AES and patent rights
Date: Tue, 29 Sep 1998 20:22:14 GMT
From: "Douglas A. Gwyn" <DAGwyn@null.net>
Message-ID: <3611414B.5D6A5474@null.net>
References: <3610DD10.7753@ssd.bna.boeing.com>
Newsgroups: sci.crypt
Lines: 63

Joseph K. Nilaad wrote:
> > Things that aren't published just aren't recognised by the community.
> Are you referring to government sponsored community?

No, he's talking about the public workers in a field,
the same "peers" that review articles submitted for
publication in refereed journals.

The issue isn't whether or not you can access the
document on-line, it's whether or not the document
has made it through a reasonable "antijunk" filter.

> To me, recognition is majority of public
> acceptance. For example, Windows was not endorsed by any organization;
> however, the majority of public accept it.

You ought to choose better examples. Windows is an
atrocious excuse for an operating system, although
we use it anyway. If operating systems had to pass
a peer-review process (one not confined to the
developing organization) before they were released
for public consumption, they might be a whole lot
better.

40 million Frenchmen *can* be wrong (and usually are).

> Is this anamoly? Isn't that normally before you create something you
> must have some procedures in mind how about doing it first?

Unfortunately, there's no good body of engineering
theory for cryptosystem design. There *is* much
knowledge about it, but an essential step is still
to turn the design over to the cryptanalysts to see
what attacks they are able to devise.

> How can you get *everyone* to TRIED to break a given cipher? if "no one
> has broken a cipher", it is secure! Think about it.

No, that is a horrible error. If nobody *can* break a cipher,
it is secure. Just because you don't happen to know of a way
to break it doesn't mean that nobody does, nor that a successful
attack won't be devised in the near future.

> 1. When NIST said that AES is to be "free world wide", will NIST make
>   exception to export law which limit to 40 bits?

NIST doesn't make US export law.
AES is intended for US federal government (unclassified) use only.

> 2. How will NIST deal with key escrow stuffs or will there be any key
>   escrow issue?

Key escrow capability is not a requirement for AES.
Presumably, the key management system would retain
a record of keys (remember: US government use) if
escrow is desired.

> 3. Where is available site to find out about AES candidates?

http://csrc.nist.gov/encryption/aes/aes_home.htm

I found that in less than a minute using a Web search engine;
try it sometime.

Subject: Re: AES and patent rights
Date: Wed, 30 Sep 1998 20:44:28 -0500
From: "Stephen M. Gardner" <gardner@metronet.com>
Message-ID: <3612DE7C.94888555@metronet.com>
References: <3610DD10.7753@ssd.bna.boeing.com>
Newsgroups: sci.crypt
Lines: 14

Joseph K. Nilaad wrote:
> How can you get *everyone* to TRIED to break a given cipher? if "no one
> has broken a cipher", it is secure! Think about it.
There is a word missing from your sentence: "yet". And when something
is not broken yet and it hasn't had a lot of attempts made yet it is
really not secure. Insecurity in a cipher seems a lot like bugs in
software. You can't prove they are not there but the more people bang
on it the more bug free you can assume it is.

--
Take a walk on the wild side: http://www.metronet.com/~gardner/
Still a lot of lands to see but I wouldn't want to stay here,
it's too old and cold and settled in its ways here.
Joni Mitchell ("California")

Subject: Re: AES and patent rights
Date: Mon, 28 Sep 1998 13:52:41 GMT
From: phr@netcom.com (Paul Rubin)
Message-ID: <phrEzzyJt.1rH@netcom.com>
References: <jgfunj-2809980053280001@207.22.198.201>
Newsgroups: sci.crypt
Lines: 18

In article <jgfunj-2809980053280001@207.22.198.201>,
W T Shaw <jgfunj@EnqvbSerrGrknf.pbz> wrote:
>> Designs are dime a dozen, so it's hard to seperate the good ones from
>> the silly ones. Good cryptanalysis is hard; it will force people to
>> take notice of your work.
>
>The big word generally preached as gospel for ages has that the world
>would be your oyster if anyone could come up with a really good
>algorithm.

I don't remember ever hearing anything like that. What I remember is
hearing that it's hard for a cipher designer to be taken seriously
until they've gotten some serious, interesting results at breaking
other people's algorithms.

>Things have changed, it seems....the welcome wagon is no longer out.

I don't see that anything has changed.

Subject: Re: AES and patent rights
Date: Mon, 28 Sep 1998 10:48:54 -0600
From: jgfunj@EnqvbSerrGrknf.pbz (W T Shaw)
Message-ID: <jgfunj-2809981049090001@207.101.116.71>
References: <phrEzzyJt.1rH@netcom.com>
Newsgroups: sci.crypt
Lines: 44

In article <phrEzzyJt.1rH@netcom.com>, phr@netcom.com (Paul Rubin) wrote:

> In article <jgfunj-2809980053280001@207.22.198.201>,
> W T Shaw <jgfunj@EnqvbSerrGrknf.pbz> wrote:
> >
> >The big word generally preached as gospel for ages has that the world
> >would be your oyster if anyone could come up with a really good
> >algorithm.
>
> I don't remember ever hearing anything like that. What I remember is
> hearing that it's hard for a cipher designer to be taken seriously
> until they've gotten some serious, interesting results at breaking
> other people's algorithms.

Over the years, it is something I have heard from many, that were
searching for real breakthroughs. Seeking to discourage, to limit such
endeavors even today, is a rather snobbish response, not very scientific,
more a political statement in line with old governmental policy:

The official line was that you had bettter not even try; if you wanted to
think in the area, you had better register; and don't think of
implementing anything without going through some sort of Greek Debate on
its utility first.

Fortunately, official propaganda did not conclusively work; it merely
translated the fears of the government into something to disregard as it

went hard astern to the obvious problem it saw.
>
> >Things have changed, it seems....the welcome wagon is no longer out.
>
> I don't see that anything has changed.

Well, I'll put a fresh coat of paint on it, publish your new ideas
here....be brave....be clear....be straightforward.....and, don't send 'em
where they are not appreciated.
--

Are you tired, rundown, can't find a corner in the office to hide in?

Then, try Jimmy Carter's Little Pills, which are apt to cause you to
want to get out your frustrations *constructively*, but might tend
to make you fear rabbits.

Decrypt with ROT13 to get correct email address.

Subject: Re: AES and patent rights
Date: Tue, 29 Sep 1998 04:59:00 GMT
From: "Douglas A. Gwyn" <DAGwyn@null.net>
Message-ID: <361068F1.EDD53427@null.net>
References: <jgfunj-2809980053280001@207.22.198.201>
Newsgroups: sci.crypt
Lines: 30

W T Shaw wrote:
> (Bruce Schneier) wrote:
> > Publication does not mean self-publication on a website,
> > it means publication in a workshop, conference, or journal.
> We are at a great transition where actual printed matter is fastly being
> usurped by electronic media...it's in all the *papers*. Holding to the
> old standard as the one true path is merely quaint.

There is something in both points of view.
The essential difference is between *peer review* and lack of review.
There undoubtedly is some good unreviewed Web publication,
as well as a lot of bogosity.
And there are also dubious or even bogus peer-reviewed articles.
Bruce's belief, shared by many, seems to be that peer review
on the average contributes to the quality (thus utility) of
the publication.

> To demand anyone must break an old algorithm to be noticed as a hurdle is
> too much given the effort required to preform feats and the sparce nature
> of such challenges. You might spend lots of effort trying to break one
> that would surpass all others as well. It is rather tempting to try to
> make algorithms that could be broken, just to increase the supply, and
> therefore qualify more people into a higher realm.

I think the intent was to not waste people's time with arbitrary
claims, but only with claims (for security) that have some
reasonable chance of being true. *One* (not the only) way to
get cryptographers to pay attention to your new cryptosystem
design is to have developed a reputation as one who understands
where other designs have been vulnerable.

Subject: Re: AES and patent rights
Date: 29 Sep 1998 22:59:48 GMT
From: an096@yfn.ysu.edu (David A. Scott)
Message-ID: <6urop4$8am$1@news.ysu.edu>
References: <jgfunj-2809980053280001@207.22.198.201>
Newsgroups: sci.crypt
Lines: 45

In a previous article, juola@mathcs.duq.edu (Patrick Juola) says:

>In article <3610DD10.7753@ssd.bna.boeing.com>,
>Joseph K. Nilaad <jknilaad@ssd.bna.boeing.com> wrote:
>>Bruce Schneier wrote:
>>>
>>> On Mon, 28 Sep 1998 00:53:27 -0600, jgfunj@EnqvbSerrGrknf.pbz (W T
>>> Shaw) wrote:
>>>
>>> >
>>> >Publication means, amongst other things, "3. Communication of information
>>> >to the public."
>>> >
>>> >We are at a great transition where actual printed matter is fastly being
>>> >usurped by electronic media...it's in all the *papers*. Holding to the
>>> >old standard as the one true path is merely quaint.
>>>
>>> The difference is not "dead trees versus electrons." The difference
>>> is "recognised publication versus self publication." If random person
>>> X puts some kind of crypto design paper on his website, almost no one
>>> in the community will read it. Yes, it is available to them. Yes,
>>> they have the technical ability to read it. But they have no idea if
>>> it is a waste of time to read or not.
>>If it is good for the goose, it's also good for the gander. Do you
>>think *everybody* will read all the works published *on paper* by NIST?

>>Beside paper copies may not be free and take too long to be delivered.
>
>Well, no. Because NIST has something that most people don't -- to
>wit, credibility. The reason that NIST has credibility is because of
>something else that NIST has that most people haven't -- technical
>expertise and knowledge of the subject.
>
That is why the NSA is trying to use them for a front. To fool
people into thinking that the contest was above board. But very
little in crypto in this country and else where is not poisioned
by the self serving NSA.
Maybe the letter NSA really stands for Nazi's Sink America
after all they got a german front man in the form of B.S.
Bruce.

--
http://cryptography.org/cgi-bin/crypto.cgi/Misc/scott19u.zip
for the version with a real key of voer one million bytes.
also scott16u.zip and scott4u.zip

Subject: Re: AES and patent rights
Date: Tue, 29 Sep 1998 20:32:20 -0600
From: jgfunj@EnqvbSerrGrknf.pbz (W T Shaw)
Message-ID: <jgfunj-2909982032210001@dialup163.itexas.net>
References: <6urop4$8am$1@news.ysu.edu>
Newsgroups: sci.crypt
Lines: 34

In article <6urop4$8am$1@news.ysu.edu>, an096@yfn.ysu.edu (David A. Scott)
wrote:

> Maybe the letter NSA really stands for Nazi's Sink America...

In various contests with the Russians, it was a question of whose captured
Germans were better.....but, that is beside the point that your comment is
really offensive.

> after all they got a german front man in the form of B.S.
> Bruce.
>
Now, your getting personal as I was called by those initials by my ag
teacher in high school, admittedly for good reason. As a matter of fact,
I'm a good bit German in heritage myself, but have no sympathy for
irrational governmental behavior regardless of where or when it
occured.... So, I find your comments here also offensive.

I saw recently Bruce impressively in action in ways I did not appreciate
before, not that I automatically now agree with him in all major areas.
Yep, he is a sympathetic favorite in some circles, but I see so many in
NIST going out of their way to be fair even if it hurts for as long as
their reins will let them run that way.

This whole business is far from simple, darn it. But, that is the appeal
of the multiact play, never knowing what will happen the next time the
lights go up. Only a rough temporary outline is in hand, and the players
will create their own roles as they go along.
--

Show me a politician who does not lie through his teeth,
and.....I'll show you one who can't find his dentures.

Decrypt with ROT13 to get correct email address.

Subject: Re: AES and patent rights
Date: Wed, 30 Sep 1998 16:03:47 GMT
From: jsavard@tenMAPSONeerf.edmonton.ab.ca (John Savard)
Message-ID: <3612534d.3078640@news.prosurfr.com>
References: <6urop4$8am$1@news.ysu.edu>
Newsgroups: sci.crypt
Lines: 39

an096@yfn.ysu.edu (David A. Scott) wrote, in part:

>Maybe the letter NSA really stands for Nazi's Sink America
>after all they got a german front man in the form of B.S.
>Bruce.

Well, the NSA has a proud history which includes sinking a lot of
Nazis (and not a few personnel of Imperial Japan as well) - and, for
that matter, I'm surprised you didn't drag in Phil Zimmerman, Terry
Ritter, or, better yet, Horst Feistel - whose progress towards U.S.
citizenship was interrupted by World War II, making him an "enemy
alien" for a few years; the others' ancestors could have come over on
the Mayflower or soon thereafter AFAIK.

Although I am unwilling to search for the appropriate words to
describe the offensiveness of this particular comment, I'm not
surprised, given the cautious and conservative tone of his most famous
work, that you harbor such sentiments: as I said in a recent post,

>of course, there are other people who claim that DES, IDEA, Blowfish,
>and all the other well-known block cipher designs are horribly

>insecure, and suggest that instead we should go and use block ciphers
>with key-dependent S-boxes with 65,536 entries in them, or Genuine
>Artificial Imitation One-Time Pads, as the only true road to security.

>Obviously, reading your book _Applied Cryptography_ will lead people
>to suspecting that you are one of the members of this "conspiracy" as
>well.

to which you replied, IIRC, noting that Scott16u and the others allow
the use of keys of different sizes.

Why you consistently fail to realize that ever more spectacular
displays of your ignorance and stupidity are not the way to convince
people that cipher designs which you have produced merit serious
consideration ... is one of life's little mysteries.

John Savard
http://members.xoom.com/quadibloc/index.html

Subject: Re: AES and patent rights
Date: 28 Sep 1998 14:26:45 GMT
From: rreynard@aol.com (RREYNARD)
Message-ID: <19980928102645.11011.00000390@ngol07.aol.com>
References: <360ed738.1402804@news.visi.com>
Newsgroups: sci.crypt
Lines: 60

In article <360ed738.1402804@news.visi.com>, schneier@counterpane.com (Bruce
Schneier) writes:

> DES was a US standard. It was free for use in the US.
>Again, if you know of anything IBM got from DES besides publicity,
>please let me know.
>
>
Off hand, I would guess that IBM got to avoid being sued for constraint of
trade.

I worked for IBM at the time they introduced DES as an option (free). It was
pretty much a "no interest" item at the time since very few of the many
computer installations needed such a capability. There was very little TP
(teleprocessing) and the main concern for data security was for back-up
purposed not so much protection from theft. I believe IBM included it as an
option to be able to sell their wares to those very few companies that required
it. There was a bit of R&D involved also that indicated that there would be a
greater need later on as more and more systems went "on-line." Probably 99% of
the marketing force "forgot" about DES even being available 24 hours after it
was announced.

The is an old saying - "Invent a better mousetrap and the world will beat a
is the Mother of Invention."

I don't begin to understand the the "problems" of providing secure
communication using cryptography so it is impossible for me to really
appreciate this exchange of opinions about "free" vs patented/copyrighted
cryptographic algorithms and systems. However, as a retired businessman, it
appears that the creator/inventor of cryptographic algorithms and systems finds
himself in the unfortunate position of being in the wrong business at the wrong
time.

The manner in which cryptography has evolved has created a situation in which
there seems to be very little demand for the "product." I think a thorough
market analysis would show that it has little or no chance to be a profitable
venture.

What does appear to have profit potential is the sale of crytographic system
implementation. If I may, an analogy - Rather than spend time and effort trying
to invent an new and "better" tool for the cryptographic toolbox, which seems
filled with more than are needed, it would seem that the better course for the
cryptanalyst would be to direct his expertise into the use of the tools and the
creation of systems using those tools as a way to make a living.

It doesn't seem fair, that a person who knows more than most about "mouse
traps" and can create better "mouse traps" than anyone else should not be able
to "profit" from this capability. Unfortunately, the marketplace is not always
fair, but it is nearly always "Darwinistic."

Timing is extremely important and this seems to be a time when the "mouse
problem" is under control (another analogy) and it is but a small part of a
much larger "pest control" problem. Therefore my "make a profit" advice would
be to stop inventing "better" mousetraps and start selling "pest control"
capability. My "have a good time and do interesting things" advice would be to
work on "mousetrap technology."

Subject: Re: AES and patent rights
Date: Mon, 28 Sep 1998 12:20:08 -0400
From: Jerry Leichter <leichter@smarts.com>
Message-ID: <360FB738.290D@smarts.com>
References: <360ebd79.18691713@news.io.com>

Newsgroups: sci.crypt
Lines: 76

A minor sub-point:

| >There is some caselaw on the subject. NIST will make a public call
| >to all third parties to state any potential patent claims regarding
| >the submissions. If someone chooses not to, NIST could argue in
| >court that the patentholder deliberately withheld information in an
| >attempt to hide his rights until after AES was awarded. Will this do
| >anygood? No one knows.
|
| As far as I know, there is *no* responsibility in patents to take
| offensive action at any particular time *or* to respond to
| governmental calls for clarification. Perhaps you are thinking of
| copyright.

There's no such requirement in copyright law either. There are a couple of areas of law in which such a requirement exists; trademarks, bank accounts, and real estate come to mind. The last has a very long history, and is quite precisely defined: If you own real estate, and someone else "visibly" - to your knowledge - uses it without your permission, and you do nothing about that use for some number of years (defined by state law - typically around 20 years), the real estate is his. (That's why the owners of private streets or pieces of sidewalk normally left open to the public close them for a day every every couple of years.)

Bank accounts that haven't been accessed in some period of time are publically listed. If the owner doesn't respond, the state ultimately takes the money. (The timeout here has gotten shorter in some states in recent years as legislators have found this a quick way to get some money into state coffers. I think it may be as short as 7 years in some states.)

For trademarks, if you don't defend you claim, you lose it. However, there's no fixed time limit - you have to show that the term hasn't slipped into common, generic usage.

These are specific special cases. There are all kinds of other "statutes of limitation" which start running as soon as you become aware that you can raise a legal claim. These limits are in the 3 year range, though there are variations for particular kinds of cases. As far as I know, there is no special limitation for patent lawsuits, beyond general statutes of limitation on civil claims. An interesting distinction here is that you may not be able to recover for *old* infringements, but still be able to recover for more recent - and ongoing - ones. This showed up in some recently-decided patent case - I can't recall the details.

However, there's a general legal principle, called I think laches, which basically says that if you arbitrarily delay raising a legal claim for too long, you may lose your right to do so. There's no black and white definition of what "too long" is in this case; it's probably stated in terms like "unreasonable delay".

"Submarine" patents - patents that surface only after some invention has been practiced, apparently without problem, for a substantial period of time - are viewed as a "bug" in the current patent system, not a feature. Remember that the constitutional basis for the existence of patents is in a trade-off: In return for making public how your invention works, you get exclusive rights to it for some number of years. Waiting for others to re-invent what you have invented, then use it for a while - and only then come along and try to take the profits - is a corruption of the intent of the system.

If, indeed, NIST aggressively searches existing patents for potential infringements, and makes a very public call for anyone who thinks they can claim infringement to come forward, then keeping your patent (well, likely pending patent) secret and trying to collect on it later would be a very risky strategy. You'd probably be viewed as coming to court with "dirty hands", and there's likely plenty of room in existing law for a judge to find some way to throw your case out.

Of course, as with all things not based on established precendent, no one can ever be really sure until the case is tried and all the appeals are heard. You should have little trouble finding lawyers willing - for a suitable fee - to argue either side. :-)

-- Jerry

Subject: Re: AES and patent rights
Date: Tue, 29 Sep 1998 07:11:00 GMT
From: ritter@io.com (Terry Ritter)
Message-ID: <361087a8.26684962@news.io.com>
References: <360FB738.290D@smarts.com>
Newsgroups: sci.crypt,misc.int-property
Lines: 77

misc.int-property added, since that is where patent lawyers hang out.

This thread concerns the competition being held by the US National Institute of Standards and Technology (NIST), an agency of Commerce.

The competition is to select a standard cipher to replace the US Data Encryption Standard (DES) with an Advanced Encryption Standard (AES).

Entry is now closed, but the competition will proceed for next several years. The issue in contention here is that all those who wished to participate in AES were required to give up any patent rights they had on their ciphers, so that NIST could make AES available "free worldwide."

On Mon, 28 Sep 1998 12:20:08 -0400, in <360FB738.290D@smarts.com>, in sci.crypt Jerry Leichter <leichter@smarts.com> wrote:

>[...]
>| As far as I know, there is *no* responsibility in patents to take
>| offensive action at any particular time *or* to respond to
>| governmental calls for clarification. Perhaps you are thinking of
>| copyright.

>[...]
>However, there's a general legal principle, called I think laches, which
>basically says that if you arbitrarily delay raising a legal claim for
>too long, you may lose your right to do so. There's no black and white
>definition of what "too long" is in this case; it's probably stated in
>terms like "unreasonable delay".

Yes. Thank you for re-acquainting me with laches.

>[...]
>If, indeed, NIST aggressively searches existing patents for potential
>infringements, and makes a very public call for anyone who thinks they
>can claim infringement to come forward, then keeping your patent (well,
>likely pending patent) secret and trying to collect on it later would be
>a very risky strategy. You'd probably be viewed as coming to court with
>"dirty hands", and there's likely plenty of room in existing law for a
>judge to find some way to throw your case out.

First let me remind everyone that I am no sort of lawyer at all, let alone a patent lawyer.

However, I would argue that we have various problems asserting laches in the present situation, and here are the most obvious to me:

1) A patent is an offensive right, yes, but it is a right to collect damages. To be asserted, there should *be* damages. But *looking* at a patented cipher is not (or may not be) damage. In fact, damage probably will not start to accrue until after the standard is set and manufacturers use it in volume. I would see *that* as the appropriate time to assert patent rights on AES. Indeed, until a particular cipher is picked, there is likely to be little or no damage at all.

2) I have essentially argued many times that the government does not enter this process with clean hands. The government has specifically worked to prevent those who will not give up their rights from having a place in the AES process. Not satisfied with that, the government now asks patent holders to come forth and be recognized for the explicit purpose of *excluding* any submitted cipher which may be covered by patents. But I argue that a patent holder has no responsibility to assist the government in abrogating his own properly-granted patent rights.

>Of course, as with all things not based on established precendent, no
>one can ever be really sure until the case is tried and all the appeals
>are heard. You should have little trouble finding lawyers willing - for
>a suitable fee - to argue either side. :-)

Terry Ritter ritter@io.com http://www.io.com/~ritter/
Crypto Glossary 1998-08-27: http://www.io.com/~ritter/GLOSSARY.HTM

Subject: Re: AES and patent rights
Date: Tue, 29 Sep 1998 16:06:11 GMT
From: jsavard@tenMAPSONeerf.edmonton.ab.ca (John Savard)
Message-ID: <361102c0.2440117@news.prosurfr.com>
References: <361087a8.26684962@news.io.com>
Newsgroups: sci.crypt,misc.int-property
Lines: 37

ritter@io.com (Terry Ritter) wrote, in part:

>Not satisfied with that, the government
>now asks patent holders to come forth and be recognized for the
>explicit purpose of *excluding* any submitted cipher which may be
>covered by patents. But I argue that a patent holder has no
>responsibility to assist the government in abrogating his own
>properly-granted patent rights.

One certainly would think that, should an AES submission be chosen that infringes on a patent, the patent-holder would not be considered negligent in enforcing his patent merely because he did not act until an infringment actually took place - someone implemented the algorithm

commercially.

Plus, if waiting for someone with deep pockets to sue constitutes an invalidation of patent rights, then so would waiting for someone to sue with small pockets for hiring a lawyer with...and there was a case a few years ago where recording companies sued a barber for having a radio on in his barber shop, in an effort to establish a new precedent on the commercial use of music.

Strictly speaking, however, this isn't assisting the government in "abrogating" patent rights, but assisting it to respect them, by using some other algorithm.

Several of the AES candidates may be infringing the IDEA patent, _if its claims are broadly interpreted_, since several of them improve their security by making use of both addition and XOR in certain places. (My use of an S-box in between in Quadibloc II may help avoid this problem...) Does MARS, with a variable-length shift, come into conflict with the RC5 patent as well? The algorithms don't, however, appear to raise any issues unless it is possible to patent the use of very basic arithmetic operations in cryptography.

John Savard
http://members.xoom.com/quadibloc/index.html

Subject: Re: AES and patent rights
Date: Tue, 29 Sep 1998 19:52:42 GMT
From: schneier@counterpane.com (Bruce Schneier)
Message-ID: <36113a43.20286586@news.visi.com>
References: <361102c0.2440117@news.prosurfr.com>
Newsgroups: sci.crypt,misc.int-property
Lines: 23

On Tue, 29 Sep 1998 16:06:11 GMT, jsavard@tenMAPSONeerf.edmonton.ab.ca
(John Savard) wrote:
>*Several* of the AES candidates may be infringing the IDEA patent, _if
>its claims are broadly interpreted_, since several of them improve
>their security by making use of both addition and XOR in certain
>places.

Interesting comment. I will look at the IDEA patent.

>(My use of an S-box in between in Quadibloc II may help avoid
>this problem...) Does MARS, with a variable-length shift, come into
>conflict with the RC5 patent as well?

Both companies seem to think that this is a possibility. IBM made
pains to talk about prior art in their submission. The RC5 patent,
though, explicitly mentions that prior art. This is something that
must be resolved

Bruce
****
Bruce Schneier, President, Counterpane Systems Phone: 612-823-1098
101 E Minnehaha Parkway, Minneapolis, MN 55419 Fax: 612-823-1590
Free crypto newsletter. See: http://www.counterpane.com

Subject: Re: AES and patent rights
Date: Tue, 29 Sep 1998 20:08:36 -0600
From: jgfunj@EnqvbSerrGrknf.pbz (W T Shaw)
Message-ID: <jgfunj-2909982008360001@dialup163.itexas.net>
References: <36113B71.1FE7EFEB@null.net>
<19980929085820.04701.00001110@ngol08.aol.com>
<361087a8.26684962@news.io.com>
Newsgroups: sci.crypt
Lines: 23

In article <36113B71.1FE7EFEB@null.net>, "Douglas A. Gwyn"
<DAGwyn@null.net> wrote:

> RREYNARD wrote:
> > This sounds very much like "crypto" COBOL and I would think has about
the same
> > chance of achieving it's objective.
>
> Could you explain that? Your reasoning is not at all obvious.
> It appears to me that, unless *all* 15 AES submissions turn out
> to have significant flaws, one of them is bound to be selected
> and be mandated as the new FIPS for unclassified encryption.
> That would be the same status that DES had.

The government reserves the right to refuse to approve a standard based on
the entries and adopt something else, or not set any standard...and not
tell why.
--

Show me a politician who does not lie through his teeth,
and.....I'll show you one who can't find his dentures.

Decrypt with ROT13 to get correct email address.

Subject: Re: AES and patent rights
Date: 27 Sep 1998 05:18:35 GMT
From: lamontg@bite.me.spammers
Message-ID: <6ukhrb$92g$1@nntp6.u.washington.edu>
References: <360d30d2.9039808@news.io.com>
Newsgroups: sci.crypt
Lines: 39

ritter@io.com (Terry Ritter) writes:
>>Whatever, it seems like
>>the idea was a good one. As I said before, we have fifteen
>>submissions, some of them very good.
>
>The competition is being conducted in a way which I believe is
>unconstitutional, which means that the result -- whatever it is --
>will be open to challenge.

On what constitutional basis? The fact that you can't make money off of
it does not imply that it is unconstitutional.

>More than that, these rules act to restrict the long term development
>of crypto technology by not allowing fundamentally-new technology to
>compete,

1. Is it a good thing for fundamentally-new technology to compete? Feistel
    ciphers have the benefit of an awful lot of cryptanalysis. I wouldn't
    want the AES to be broken 5 years later when everyone goes "oops, i guess
    that new approach *wasn't* all that good."

2. Fundamentally-new technology is not banned from competition. The
    submitters simply have to accept other forms of compensation (e.g. PR)
    rather than obtaining a guaranteed lump of money from the government
    after their cipher design is mandated in all government contracts.

>and by not rewarding the crypto design process itself. These
>rules are tools to minimize the open development of cryptographic
>technology, and every entrant who participates is another government
>argument that this is a good thing.

If *ANYTHING* is unconstitutional and un-american it is giving the winner
of this contest a guaranteed 15 or so year contract to reap royalties from
every government-used encryption product out there. It's like saying that
all the cars the government uses for the next 15 years will be Fords.

--
Lamont Granquist (lamontg@u.washington.edu)
looking for unix administration / security work

Subject: Re: AES and patent rights
Date: Sun, 27 Sep 1998 08:22:42 GMT
From: ritter@io.com (Terry Ritter)
Message-ID: <360df5b4.17036475@news.io.com>
References: <6ukhrb$92g$1@nntp6.u.washington.edu>
Newsgroups: sci.crypt
Lines: 81

On 27 Sep 1998 05:18:35 GMT, in <6ukhrb$92g$1@nntp6.u.washington.edu>,
in sci.crypt lamontg@bite.me.spammers wrote:

>ritter@io.com (Terry Ritter) writes:
>>The competition is being conducted in a way which I believe is
>>unconstitutional, which means that the result -- whatever it is --
>>will be open to challenge.
>
>On what constitutional basis? The fact that you can't make money off of
>it does not imply that it is unconstitutional.

Perhaps if you would read more, and delete less, the answer would
become apparent:

(from that same article)
>>My argument would more likely be based on "equal protection under
>>the law" than antitrust; when government participation is limited
>>by what one owns, there have to be questions. There is also "taking
>>without compensation."

>>More than that, these rules act to restrict the long term development
>>of crypto technology by not allowing fundamentally-new technology to
>>compete,
>
>1. Is it a good thing for fundamentally-new technology to compete? Feistel
>   ciphers have the benefit of an awful lot of cryptanalysis. I wouldn't
>   want the AES to be broken 5 years later when everyone goes "oops, i guess
>   that new approach *wasn't* all that good."

Competition is competition. For Feistel technology to be "better," it
must survive in competition. If it can't, maybe Feistel technology is
not as much "better" as we thought it was.

>2. Fundamentally-new technology is not banned from competition. The
>   submitters simply have to accept other forms of compensation (e.g. PR)
>   rather than obtaining a guaranteed lump of money from the government
>   after their cipher design is mandated in all government contracts.

I do *not* "have to accept" "other forms of compensation."

Personally, I would have been willing to give *the government* a very
attractive license. But that is not enough for our government: AES
demanded that I give a free license to every for-profit company who
would be *selling* software based on that technology. That sure
sounds like a government-mandated subsidy for those guys, doesn't it?

Then I would get to compete with those guys, "on an equal footing,"
after they used *their* resources getting ahead in other areas, and
with *no* free help from them. Yeah, that sounds fair.

>>and by not rewarding the crypto design process itself. These
>>rules are tools to minimize the open development of cryptographic
>>technology, and every entrant who participates is another government
>>argument that this is a good thing.
>
>If *ANYTHING* is unconstitutional and un-american it is giving the winner
>of this contest a guaranteed 15 or so year contract to reap royalties from
>every government-used encryption product out there. It's like saying that
>all the cars the government uses for the next 15 years will be Fords.

MULTIPLE CHOICE:

If Ford is the winner in a contest for the most-efficient car design,
based on long-term, very expensive and privately-funded research,
would we expect:

a) they should be made to give their superior technology to all their
competitors, free of charge, or

b) they should reap just rewards for their successful research.

Terry Ritter ritter@io.com http://www.io.com/~ritter/
Crypto Glossary 1998-08-27: http://www.io.com/~ritter/GLOSSARY.HTM

Subject: Re: AES and patent rights
Date: 27 Sep 1998 23:02:14 GMT
From: lamontg@bite.me.spammers
Message-ID: <6umg5m$19ke$1@nntp6.u.washington.edu>
References: <360df5b4.17036475@news.io.com>
Newsgroups: sci.crypt
Lines: 113

ritter@io.com (Terry Ritter) writes:
>On 27 Sep 1998 05:18:35 GMT, in <6ukhrb$92g$1@nntp6.u.washington.edu>,
>in sci.crypt lamontg@bite.me.spammers wrote:
>>ritter@io.com (Terry Ritter) writes:
>>>The competition is being conducted in a way which I believe is
>>>unconstitutional, which means that the result -- whatever it is --
>>>will be open to challenge.
>>
>>On what constitutional basis? The fact that you can't make money off of
>>it does not imply that it is unconstitutional.
>
>Perhaps if you would read more, and delete less, the answer would
>become apparent:
>
>(from that same article)
>>>My argument would more likely be based on "equal protection under
>>>the law" than antitrust; when government participation is limited
>>>by what one owns, there have to be questions.

Perhaps if you would clarify then it would become apparent. Explain how
you think that equal protection applies.

>>>There is also "taking
>>>without compensation."

Which is silly. The government isn't taking anything which isn't freely
offered.

>>>More than that, these rules act to restrict the long term development
>>>of crypto technology by not allowing fundamentally-new technology to
>>>compete,
>>
>>1. Is it a good thing for fundamentally-new technology to compete? Feistel
>>   ciphers have the benefit of an awful lot of cryptanalysis. I wouldn't
>>   want the AES to be broken 5 years later when everyone goes "oops, i guess
>>   that new approach *wasn't* all that good."
>
>Competition is competition. For Feistel technology to be "better," it
>must survive in competition. If it can't, maybe Feistel technology is
>not as much "better" as we thought it was.

Well, it seems to be quite adequately surviving in the non-AES marketplace
just fine.

>>2. Fundamentally-new technology is not banned from competition. The
>>   submitters simply have to accept other forms of compensation (e.g. PR)
>>   rather than obtaining a guaranteed lump of money from the government
>>   after their cipher design is mandated in all government contracts.
>
>I do *not* "have to accept" "other forms of compensation."

Yes, you do because that's how the AES competition is set up.

>Personally, I would have been willing to give *the government* a very
>attractive license.

Yes, and everyone else would have to license from you, which would inhibit
the implimentation of the algorithm in other forms of software. If it is
to be widely used, it should be free.

>But that is not enough for our government: AES
>demanded that I give a free license to every for-profit company who
>would be *selling* software based on that technology.

No, NIST is demanding that whoever wins the competition give free license
to anyone who wants to use it -- irregardless of if they are selling or
not. They are not demanding that you give a free license to anything -- just
don't enter the competition.

>That sure
>sounds like a government-mandated subsidy for those guys, doesn't it?

And it also allows every not-for-profit company or person to write AES
software and distribute it unencumbered, as well. If it's a subsidy then it's
your subsidy against their subsidy.

>Then I would get to compete with those guys, "on an equal footing,"
>*after* they used *their* resources getting ahead in other areas, and
>with *no* free help from them. Yeah, that sounds fair.

Then don't enter the competition. Simple.

>>>and by not rewarding the crypto design process itself. These
>>>rules are tools to minimize the open development of cryptographic
>>>technology, and every entrant who participates is another government
>>>argument that this is a good thing.
>>
>>If *ANYTHING* is unconstitutional and un-american it is giving the winner
>>of this contest a guaranteed 15 or so year contract to reap royalties from
>>every government-used encryption product out there. It's like saying that
>>all the cars the government uses for the next 15 years will be Fords.
>
>MULTIPLE CHOICE:
>
>If Ford is the winner in a contest for the most-efficient car design,
>based on long-term, very expensive and privately-funded research,
>would we expect:
>
>a) they should be made to give their superior technology to all their
>competitors, free of charge, or
>
>b) they should reap just rewards for their successful research.

b. In this case "just rewards" are the PR and cipher design expertise which
is entirely sufficient for counterpane and apparently sufficient for the 15
other applicants. It doesn't mean that you can kick back for 15 years and
do nothing other than license the cipher -- is that "innovative" is that
promoting "competition" or is that one company winning the lottery and getting
to kick back on guaranteed money?

--
Lamont Granquist (lamontg@u.washington.edu)
looking for unix administration / security work

Subject: Re: AES and patent rights
Date: Mon, 28 Sep 1998 04:23:48 GMT
From: "Douglas A. Gwyn" <DAGwyn@null.net>
Message-ID: <360F0F30.65F94F17@null.net>
References: <360df5b4.17036475@news.io.com>
Newsgroups: sci.crypt
Lines: 16

Terry Ritter wrote:
> >If *ANYTHING* is unconstitutional and un-american it is giving the winner
> >of this contest a guaranteed 15 or so year contract to reap royalties from
> >every government-used encryption product out there. It's like saying that
> >all the cars the government uses for the next 15 years will be Fords.
> MULTIPLE CHOICE:
> If Ford is the winner in a contest for the most-efficient car design,
> based on long-term, very expensive and privately-funded research,
> would we expect:
> a) they should be made to give their superior technology to all their
> competitors, free of charge, or

> b) they should reap just rewards for their successful research.

The correct answer is "None of the above."

You should know better than to argue via inexact analogies.

Subject: Re: AES and patent rights
Date: Sun, 27 Sep 1998 22:40:31 GMT
From: schneier@counterpane.com (Bruce Schneier)
Message-ID: <360ebdde.481581@news.visi.com>
References: <6ukhrb$92g$1@nntp6.u.washington.edu>
Newsgroups: sci.crypt
Lines: 32

On 27 Sep 1998 05:18:35 GMT, lamontg@bite.me.spammers wrote:
>1. Is it a good thing for fundamentally-new technology to compete? Feistel
> ciphers have the benefit of an awful lot of cryptanalysis. I wouldn't
> want the AES to be broken 5 years later when everyone goes "oops, i guess
> that new approach *wasn't* all that good."

Different submitters have different ideas. The Twofish design team decided that fundamentally new technologies and ideas were too risky for a long-term standard. Other submitters thought differently. RC6, for example, is heavily dependent on modular multiplications and data-dependent rotations, two technologies that are very new and don't have a track record. CAST-256, by using an incomplete Feistel network, has opened itself to a lot of truncated diffferential attacks and the new impossible crytpanalysis. Serpent, on the other hand, was even more conservative than we were.

The best way for a new technique to gain acceptance is for it to be obviously better than the old techniques. If someone invented a form of cryptanalysis that blew everything else out of the water, and then described a technique that is resistant, everyone would take a serious look at it. And if it were patented, there would be nothing we could do.

Fundamentally new techniques for the sake of themselves have no place in a conservative standard. Fundamentally new techniques that resist attacks that break other things are much more interesting.

Bruce
****
Bruce Schneier, President, Counterpane Systems Phone: 612-823-1098
101 E Minnehaha Parkway, Minneapolis, MN 55419 Fax: 612-823-1590
Free crypto newsletter. See: http://www.counterpane.com

Subject: Re: AES and patent rights
Date: 27 Sep 1998 12:28:37 GMT
From: an096@yfn.ysu.edu (David A. Scott)
Message-ID: <6ulb1ljmr1@news.ysu.edu>
References: <N906423249.20630@ruby.ansuz.sooke.bc.ca>
<6ubqtt$245$1@quine.mathcs.duq.edu>
<360df5b4.17036475@news.io.com>
Newsgroups: sci.crypt
Lines: 58

In a previous article, ritter@io.com (Terry Ritter) says:

>
>On 27 Sep 1998 05:18:35 GMT, in <6ukhrb$92g$1@nntp6.u.washington.edu>,
>in sci.crypt lamontg@bite.me.spammers wrote:
>
>>ritter@io.com (Terry Ritter) writes:
>>>The competition is being conducted in a way which I believe is
>>>unconstitutional, which means that the result -- whatever it is --
>>>will be open to challenge.
>>
>>On what constitutional basis? The fact that you can't make money off of
>>it does not imply that it is unconstitutional.
>
>Perhaps if you would read more, and delete less, the answer would
>become apparent:
>
>(from that same article)
>>>My argument would more likely be based on "equal protection under
>>>the law" than antitrust; when government participation is limited
>>>by what one owns, there have to be questions. There is also "taking
>>>without compensation."
>
>
>>>More than that, these rules act to restrict the long term development
>>>of crypto technology by not allowing fundamentally-new technology to
>>>compete,
>>
>>1. Is it a good thing for fundamentally-new technology to compete? Feistel
>> ciphers have the benefit of an awful lot of cryptanalysis. I wouldn't
>> want the AES to be broken 5 years later when everyone goes "oops, i guess
>> that new approach *wasn't* all that good."
>
>Competition is competition. For Feistel technology to be "better," it

>must survive in competition. If it can't, maybe Feistel technology is
>not as much "better" as we thought it was.
>
>
>>2. Fundamentally-new technology is not banned from competition. The
>>   submitters simply have to accept other forms of compensation (e.g. PR)
>>   rather than obtaining a guaranteed lump of money from the government
>>   after their cipher design is mandated in all government contracts.
>
>I do *not* "have to accept" "other forms of compensation."
>
   IF they would just give me a life time pass to the MUSTANG I would
stop the spreading of my advacned crypto so the NSA would have an
easier time breaking every thing. But I don't think the US government
wants to really compensate anybodys who is not political correct and
I have a feeling that I am to honest for government any more.

--
http://cryptography.org/cgi-bin/crypto.cgi/Misc/scott19u.zip
for the version with a real key of voer one million bytes.
 also scott16u.zip and scott4u.zip

Subject: Re: AES and patent rights
Date: 27 Sep 1998 19:09:07 GMT
From: jsavard@freenet.edmonton.ab.ca ()
Message-ID: <6um2gj$du0$3@news.sas.ab.ca>
References: <360d0782.2157160@news.visi.com>
Newsgroups: sci.crypt
Lines: 21

Bruce Schneier (schneier@counterpane.com) wrote:
Terry Ritter wrote:

: >It is unfortunate that Bruce Schneier was a prime factor in getting
: >the original rules changed so that only free designs would even be
: >*considered* for AES.

: Was I? Wow. I thought that was NIST's idea. Whatever, it seems like
: the idea was a good one. As I said before, we have fifteen
: submissions, some of them very good.

Having followed the discussions leading up to the final call for AES
submissions, I know what Mr. Ritter is basing his claim on here. Initially
the AES committee was expressing a strong preference for royalty-free
submissions; this changed to a mandatory requirement shortly after a
statement by you that the preference was unlikely to create problems.

That they may have had other input, or other reasons for making that
change, certainly isn't disproven by the record.

John Savard

Subject: Re: AES and patent rights
Date: Sun, 27 Sep 1998 22:35:06 GMT
From: schneier@counterpane.com (Bruce Schneier)
Message-ID: <360ebd88.396141@news.visi.com>
References: <6um2gj$du0$3@news.sas.ab.ca>
Newsgroups: sci.crypt
Lines: 29

On 27 Sep 1998 19:09:07 GMT, jsavard@freenet.edmonton.ab.ca () wrote:

>Bruce Schneier (schneier@counterpane.com) wrote:
>Terry Ritter wrote:
>
>: >It is unfortunate that Bruce Schneier was a prime factor in getting
>: >the original rules changed so that only free designs would even be
>: >*considered* for AES.
>
>: Was I? Wow. I thought that was NIST's idea. Whatever, it seems like
>: the idea was a good one. As I said before, we have fifteen
>: submissions, some of them very good.
>
>Having followed the discussions leading up to the final call for AES
>submissions, I know what Mr. Ritter is basing his claim on here. Initially
>the AES committee was expressing a strong preference for royalty-free
>submissions; this changed to a mandatory requirement shortly after a
>statement by you that the preference was unlikely to create problems.
>
>That they may have had other input, or other reasons for making that
>change, certainly isn't disproven by the record.

Thanks for the explanation.

Bruce
****
Bruce Schneier, President, Counterpane Systems   Phone: 612-823-1098
101 E Minnehaha Parkway, Minneapolis, MN 55419   Fax: 612-823-1590
           Free crypto newsletter. See: http://www.counterpane.com

Subject: Re: AES and patent rights
Date: 27 Sep 1998 19:52:22 GMT
From: an096@yfn.ysu.edu (David A. Scott)
Message-ID: <6um51mdrq1@news.ysu.edu>
References: <N906423249.20630@ruby.ansuz.sooke.bc.ca>
<6ubqtt$245$1@quine.mathcs.duq.edu>
<6um2gj$du0$3@news.sas.ab.ca>
Newsgroups: sci.crypt
Lines: 33

In a previous article, jsavard@freenet.edmonton.ab.ca () says:

>Bruce Schneier (schneier@counterpane.com) wrote:
>Terry Ritter wrote:
>
>: >It is unfortunate that Bruce Schneier was a prime factor in getting
>: >the original rules changed so that only free designs would even be
>: >*considered* for AES.
>
>: Was I? Wow. I thought that was NIST's idea. Whatever, it seems like
>: the idea was a good one. As I said before, we have fifteen
>: submissions, some of them very good.

I really wonder which if not all of the methods are NSA fronts.
I think it is udder foolish to think the winner will not be a
NSA entry no matter how they try to denny it. The beauty of
this is more than most realize. Suppose it turns out weak.
Then they can claim they knew it all along but for security
reasins decided not to tell. That way they will recieve
no blame no matter what happens. When you stop to think about
it. It can't be any other way. They have to by there charter get
envovled in all american crypto. Or at least what little
we know about there secret charter. Of course that could
get Clinton to come on TV and deny there involvemnet in
such a scheme. I am sure that most americans still so
blinded my his skill that they would belive him.

--
http://cryptography.org/cgi-bin/crypto.cgi/Misc/scott19u.zip
for the version with a real key of voer one million bytes.
also scott16u.zip and scott4u.zip

Subject: Re: AES and patent rights
Date: Sun, 27 Sep 1998 22:35:48 GMT
From: schneier@counterpane.com (Bruce Schneier)
Message-ID: <360ebda9.428333@news.visi.com>
References: <6um51mdrq1@news.ysu.edu>
Newsgroups: sci.crypt
Lines: 28

On 27 Sep 1998 19:52:22 GMT, an096@yfn.ysu.edu (David A. Scott) wrote:

>
>In a previous article, jsavard@freenet.edmonton.ab.ca () says:
>
>>Bruce Schneier (schneier@counterpane.com) wrote:
>>Terry Ritter wrote:
>>
>>: >It is unfortunate that Bruce Schneier was a prime factor in getting
>>: >the original rules changed so that only free designs would even be
>>: >*considered* for AES.
>>
>>: Was I? Wow. I thought that was NIST's idea. Whatever, it seems like
>>: the idea was a good one. As I said before, we have fifteen
>>: submissions, some of them very good.
>
> I really wonder which if not all of the methods are NSA fronts.
>I think it is udder foolish to think the winner will not be a
>NSA entry no matter how they try to denny it.

Well, I know that Twofish is not an NSA entry, but there is no way I
can prove that to you.

Bruce
****
Bruce Schneier, President, Counterpane Systems Phone: 612-823-1098
101 E Minnehaha Parkway, Minneapolis, MN 55419 Fax: 612-823-1590
Free crypto newsletter. See: http://www.counterpane.com

Subject: Re: AES and patent rights
Date: 27 Sep 1998 23:04:35 GMT
From: lamontg@bite.me.spammers
Message-ID: <6umga3$td0$1@nntp6.u.washington.edu>
References: <6um51mdrq1@news.ysu.edu>
Newsgroups: sci.crypt
Lines: 9

an096@yfn.ysu.edu (David A. Scott) writes:

> I really wonder which if not all of the methods are NSA fronts.

Yeah, twofish was designed by the NSA, fronted by Bruce and Counterpane.
Likewise, RSADSI is another NSA front.

--
Lamont Granquist (lamontg@u.washington.edu)
looking for unix administration / security work

Subject: Re: AES and patent rights
Date: 28 Sep 1998 02:36:56 GMT
From: an096@yfn.ysu.edu (David A. Scott)
Message-ID: <6umso8$3k6$1@news.ysu.edu>
References: <N906423249.20630@ruby.ansuz.sooke.bc.ca>
<6ubqtt$245$1@quine.mathcs.duq.edu>
<360ebd79.18691713@news.io.com>
Newsgroups: sci.crypt
Lines: 91

In a previous article, ritter@io.com (Terry Ritter) says:

>
>On Sat, 26 Sep 1998 21:37:43 GMT, in <360d5983.3024744@news.visi.com>,
>in sci.crypt schneier@counterpane.com (Bruce Schneier) seemed to
>respond, yet failed to address his own analogy:
>
>>[...]
>>I don't believe that it is
>>>un-American, unconstitutional, or inappropriate for automobile
>>>companies to sponsor race cars, either.
>>
>>Really? You would force everyone who entered a car in the race to
>>sign over their rights to their design -- including any new
>>innovations -- if they won?
>>
>>That sounds like a very strange race to me.
>>
>>Race drivers and their organizations have to make real money, and they
>>depend upon the innovations in their cars. I doubt they would give up
>>their rights -- unless of course they simply *have* no rights, and so
>>take the opportunity to exclude their competition.
>>
>>Somebody might even have the balls to take something like that to
>>court. Especially if the race was government-sponsored.
>
>
>>[...]
>>From what I have been researched, IBM has never sued or even
>>threatened anyone for using DES. If you have other evidence, I very
>>much want to hear it?
>
>Please try to follow along: DES was a US standard. It was free for
>use in the US. Presumably IBM got something for that. Lawsuits and
>threatening have nothing to do with it.
>
>
>>[...]
>>Although more likely the government didn't want to force users of AES
>>to pay royalties, when there was the very strong possibility that free
>>alternatives migh be out there. So NIST took a risk in only asking
>>for unencumbered submissions, but it looks like their risk paid off.
>>You and I and everyone else who builds encryption systems using AES
>>will benefit.
>
>A standard *cipher* should be an advantage for bankers who want the
>liability protection of "due diligence."
>
>But companies and individuals can make their own decisions about what
>cipher to use, based on the opinions of experts they trust, or just
>random chance. Freedom is like that.
>
Terry I agree with you here people should be FREE to pick
what they want. The AES competation is just another clipper
chip in sheeps clothing. It would be best to let market
place decide on its own. Some will sell like microsoft and
some like GNU LINUX would be free but the free market should
decide.

>On the other hand, a government *interface* standard which could
>handle (virtually) any cipher of any sort as dynamically selected,
>would be useful.
>
>
>>>My argument would more likely be based on "equal protection under the
>>>law" than antitrust; when government participation is limited by what
>>>one owns, there have to be questions. There is also "taking without
>>>compensation."
>>
>>NIST is not taking anything without compensation. Everything is being
>>given freely. You are not being compelled to submit and to give up
>>your rights.
>

>Indeed, I did not submit.
>
>But *you* get to participate in a government-funded process which took
>nothing from you, but would take property from me.

 Actually Bruce will make a bundle out of this process. The very
nature of his position makes Bruce look like an expert and he will
sell more books. It makes him look like an expert while people like
you are cut out and forgotten. Government has a way of creating its
own experts. And the association he has with government we make the
sheep blindly follow him. Just a thought.

--
http://cryptography.org/cgi-bin/crypto.cgi/Misc/scott19u.zip
for the version with a real key of voer one million bytes.
 also scott16u.zip and scott4u.zip

Subject: Re: AES and patent rights
Date: 28 Sep 1998 02:49:55 GMT
From: an096@yfn.ysu.edu (David A. Scott)
Message-ID: <6umtgj$512$1@news.ysu.edu>
References: <N906423249.20630@ruby.ansuz.sooke.bc.ca>
   <6ubqtt$245$1@quine.mathcs.duq.edu>
   <360ebdde.481581@news.visi.com>
Newsgroups: sci.crypt
Lines: 38

In a previous article, schneier@counterpane.com (Bruce Schneier) says:

>Different submitters have different ideas. The Twofish design team
>decided that fundamentally new technologies and ideas were too risky
>for a long-term standard. Other submitters thought differently. RC6,
>for example, is heavily dependent on modular multiplications and
>data-dependent rotations, two technologies that are very new and don't
>have a track record. CAST-256, by using an incomplete Feistel
>network, has opened itself to a lot of truncated diffferential attacks
>and the new impossible crytpanalysis. Serpent, on the other hand, was
>even more conservative than we were.
>
>The best way for a new technique to gain acceptance is for it to be
>obviously better than the old techniques. If someone invented a form
>of cryptanalysis that blew everything else out of the water, and then
>described a technique that is resistant, everyone would take a serious
>look at it. And if it were patented, there would be nothing we could
>do.

  No the best way will never be found like what you described. More
like the current history of crypto that is so called new is NSA blessed
since they do not want good crypto. It will be very hard to get
good crypto to the masses when so called experts poo poo the idea
of long keys as snake oil. And the obvious use the inferior way
blocks are chained together is nothing but a bald face trick to
keep good crypto out of the hands of ordinary people. It should
be obvious what is going on but people like you are good at distorting
the truth.
 Hell you will never run a contest like I did at d o e sysworks because
if you only changed 4 chars in the file some one might break it.
If my stuff is so bad you try to break the contest. You can even
use your friends at the NSA.

--
http://cryptography.org/cgi-bin/crypto.cgi/Misc/scott19u.zip
for the version with a real key of voer one million bytes.
 also scott16u.zip and scott4u.zip

Subject: Re: AES and patent rights
Date: 28 Sep 1998 02:52:13 GMT
From: an096@yfn.ysu.edu (David A. Scott)
Message-ID: <6umtkt$587$1@news.ysu.edu>
References: <6ubqtt$245$1@quine.mathcs.duq.edu>
   <6um2gj$du0$3@news.sas.ab.ca>
Newsgroups: sci.crypt
Lines: 17

In a previous article, lamontg@bite.me.spammers () says:

>an096@yfn.ysu.edu (David A. Scott) writes:
>> I really wonder which if not all of the methods are NSA fronts.
>
>Yeah, twofish was designed by the NSA, fronted by Bruce and Counterpane.
>Likewise, RSADSI is another NSA front.
>
 I know you are jesting but TWOFISH would have my bet as NSA plant
I just am not sure which of the rest are.

--
http://cryptography.org/cgi-bin/crypto.cgi/Misc/scott19u.zip

for the version with a real key of voer one million bytes.
also scott16u.zip and scott4u.zip

Subject: Re: AES and patent rights
Date: Mon, 28 Sep 1998 01:11:20 -0600
From: jgfunj@EnqvbSerrGrknf.pbz (W T Shaw)
Message-ID: <jgfunj-2809980111210001@207.22.198.201>
References: <6umtkt$587$1@news.ysu.edu>
Newsgroups: sci.crypt
Lines: 44

In article <6umtkt$587$1@news.ysu.edu>, an096@yfn.ysu.edu (David A. Scott)
wrote:

> In a previous article, lamontg@bite.me.spammers () says:
>
> >an096@yfn.ysu.edu (David A. Scott) writes:
> >> I really wonder which if not all of the methods are NSA fronts.
> >
> >Yeah, twofish was designed by the NSA, fronted by Bruce and Counterpane.
> >Likewise, RSADSI is another NSA front.
> >
> I know you are jesting but TWOFISH would have my bet as NSA plant
> I just am not sure which of the rest are.
>
If some are merely traveling a convenient trail laid out by the
government, leading them to do things a particular way because of
established crypto practices, all would be swayed to do things in certain
conventional ways, probably some of which would be compatible with NSA
notions. However, I get the drift that some aspects of the process are
not so sympathetic to the desires in certain corners of our government.
Internal opinions vary wildly within all relevant organizations regarding
what is best.

In the various presentations, I was looking for some unusual information,
including a basic honesty, sincerity, knowledge and focus of the
presenters. In those areas, as well as others, Bruce shined. If
anything, government is probably a bit resentful of his status, and of the
gall of so many to take on things that they wished they still exclusively
controlled.

If I were looking for plants, and I'm not, I would rather look at those
who already have their hands in each others pockets. It would be hard not
to consider that a company would be beholden in some way if it were
involved extensively in cooperative areas with the government, contracts,
personel sharing, etc.
--

Are you tired, rundown, can't find a corner in the office to hide in?

Then, try Jimmy Carter's Little Pills, which are apt to cause you to
want to get out your frustrations *constructively*, but might tend
to make you fear rabbits.

Decrypt with ROT13 to get correct email address.

Subject: Re: AES and patent rights
Date: 28 Sep 1998 03:09:21 GMT
From: an096@yfn.ysu.edu (David A. Scott)
Message-ID: <6umul1$7c8$1@news.ysu.edu>
References: <N906423249.20630@ruby.ansuz.sooke.bc.ca>
   <6ubqtt$245$1@quine.mathcs.duq.edu>
   <360ed738.1402804@news.visi.com>
Newsgroups: sci.crypt
Lines: 35

In a previous article, schneier@counterpane.com (Bruce Schneier) says:
>What is AES process taking from you? You were not compelled to
>submit, so AES will not take your work away from you. I know that you
>patent your ideas, so if the eventual AES algorithm infringes on any
>of your patents then you will demand your rights. I don't see
>anything of yours being taken away.
  To be more honest. If the government ends up using your ideas. Its
a fuck you situation since you don't have the money to hire the lawyers
to win. So Bruce is wrong again and you would be the loser Terry

>
>Moo. Oops, sorry. Baaa.
  Terry if you did this kind of crap that Bruce did here people would
not like you. But Bruce has the money and power to laugh in your face
even if he knows that some of your ideas better than his. Because he
is the recognized government expert while you are a has been drip in
his eyes. Of course I am on the list to but he fears you more
since you can string english together a hell of a lot better than
me.

  Terry wake up the contest is a joke anyway. It is nothing but
another clipper chip in disgise. If enough people wake up maybe
it will die just like the stupid clipper chip. Socierty
would be better served if there are many methods out there.
Since stupid to but all eggs in one basket.

Yes it would be dumb if every one in the world used mine. But
it least I think even Bruce realizes I don't work for the NSA
I am not sure that they want free thinkers.

--
http://cryptography.org/cgi-bin/crypto.cgi/Misc/scott19u.zip
for the version with a real key of voer one million bytes.
also scott16u.zip and scott4u.zip

Subject: Re: AES and patent rights
Date: 29 Sep 1998 11:35:29 GMT
From: an096@yfn.ysu.edu (David A. Scott)
Message-ID: <6uqgm1$gc7$1@news.ysu.edu>
References: <jgfunj-2209982001430001@207.22.198.219>
<N906423249.20630@ruby.ansuz.sooke.bc.ca>
<3610a518.969320@news.erols.com>
Newsgroups: sci.crypt
Lines: 53

In a previous article, amungedtempdog@munged.see.sig (A [Temporary] Dog) says:

>On Tue, 29 Sep 1998 07:38:05 GMT, ritter@io.com (Terry Ritter) wrote:
>
>>
>>On Mon, 28 Sep 1998 13:25:01 GMT, in <360f8dcf.3312669@news.visi.com>,
>>in sci.crypt schneier@counterpane.com (Bruce Schneier) wrote:
>>
>>>[...]
>>>To me it looks like we were both given the same decision to make, and
>>>you chose one path and I chose the other. You believed that your
>>>patent rights were worth more than NIST was willing to give you for
>>>them.
>>[...]
>>Had AES offered even token real compensation for these rights, you
>>might have a point. They did not.
>
>If you really believe that the prestige of wining the AES contest is
>worth nothing, why do you care if you participate or not? If the
>prestige is worth something (to anyone), it is an offer of
>compensation. If it's worth nothing, then you have lost nothing by
>not participating. The AES contestants evidently believe that winning
>the contest is worth something to them. For some of them, prestige is
>readily convertible to cash via increased charges for consulting work,
>etc.
>
>They made an offer (prestige for algorithm). You chose not to accept
>their offer. Others did choose to accept their offer. This is an
>example of free trade. The fact that their offer of payment is in
>intangibles doesn't change that. They didn't force you to participate
>on their terms and you can't force them to participate on *your*
>terms. The fact that they are the government and not a business is
>also irrelevent; it's still an example of free trade.
>
>
Actually they most likely would not take his method in consideration
just as they would not take mine. The competetion is a fucking joke
only open to snobbish assholes like bruce him self and yes it will
rasise the opiniun of the sheep by giving more prestige to the snobbish
assholes involved. Of course the NSA horse will win. I think Bruce is
on that horse but I am sure the NSA has more than one. candidate in
that race. It would be foolish to think otherwise.
And if you are wondering yes I did try to get in but I am sure
what little corresponabce there was has long been filed in that
round circular file. So fuck the so called openness of the contest.

--
http://cryptography.org/cgi-bin/crypto.cgi/Misc/scott19u.zip
for the version with a real key of voer one million bytes.
also scott16u.zip and scott4u.zip

Subject: Re: AES and patent rights
Date: 29 Sep 1998 22:18:06 GMT
From: rreynard@aol.com (RREYNARD)
Message-ID: <19980929181806.04330.00019263@ngol01.aol.com>
References: <36113B71.1FE7EFEB@null.net>
Newsgroups: sci.crypt
Lines: 38

In article <36113B71.1FE7EFEB@null.net>, "Douglas A. Gwyn" <DAGwyn@null.net>
writes:

>RREYNARD wrote:
>> This sounds very much like "crypto" COBOL and I would think has about the
>same
>> chance of achieving it's objective.
>
>Could you explain that? Your reasoning is not at all obvious.

I rather thought that everyone who reads Sci.Crypt would have seen my post, smiled, and pressed on with the grim business of determining the future of cryptography.

But, since you asked -

As I recall, COBOL was the government's "standardized" Common Business Oriented Language to be used by all government agencies for application programming.

I don't believe it ever achieved "standardization", it never became "common" and many government agencies opted not to use it. It is my opinion, that the "standard" crypto algorithm will realize similar success.

Also, did the government programming standards committee that was responsible for COBOL have the foresight to require as a standard - an 8 position date field - and will a similar group be determining the size of the key length? :-)

While were are on the topic of "standards", I have a meter wrench set that I would like to sell that I purchased in anticipation of the "measurement standards" that the government were to adopt a few years ago. ;-)

And, just to get it all out, it would seem to me that rather than giving "it" away for free, the government should operate on the basis that whatever they decide to "use" as a standard, it should have a price tag. If those who would use it must pay for it, it is more likely to be "worth the cost." Actually, upon reflection, that is probably how it will turn out regardless. ;-)

Subject: Re: AES and patent rights
Date: 29 Sep 1998 23:38:36 GMT
From: an096@yfn.ysu.edu (David A. Scott)
Message-ID: <6urr1schf1@news.ysu.edu>
References: <N906423249.20630@ruby.ansuz.sooke.bc.ca>
<6ubqtt$245$1@quine.mathcs.duq.edu>
<3611414c.1821978@news.io.com>
Newsgroups: sci.crypt
Lines: 58

In a previous article, ritter@io.com (Terry Ritter) says:

>
>On Tue, 29 Sep 1998 21:25:52 +0200, in
><Pine.GSO.4.03.9809292117190.29627-100000@sun5>, in sci.crypt
><tbb03ar@mail.lrz-muenchen.de> wrote:
>
>>[...]
>>AES wouldn't be worth anything if it would be patented: Nobody is willing
>>to pay for an algorithm if there are lots of others in the public domain.
>
>RSA.
>
>RC4 (in the sense that it was easy to export).
>
>(Both of which are not free.)
>
>
>>To get a standard it was neccessary to find free programs.
>
>First, AES is a cipher; a major component, assuredly, but still only
>one component of a complete system. It is not a "program."
>
>And while there may be some "free" programs which use AES, we can be
>sure that commercial software firms will compensate their programmers
>by charging for the software. Programmers thus will be compensated --
>and justly so -- for the time they spend; but cipher designers will
>*not* be compensated for the vastly greater time *they* spend. And
>though I do wear both hats, I still find this irritating, since it is
>a direct result of government action.
>
>
>>[...]
>>BTW: Do you think the development of GNU C is unfair against Borland and
>>Microsoft?
>
>I guess that would first depend upon whether the government was
>supporting GNU C, and next whether the government would be
>recommending GNU C and even requiring it for their own use.

I think most management mistakenly thinks that you have to pay lots of money to get something good. When I worked for the government we got the people at are site to use GNU C which produces faster running code than the SUN compiler. Upper management being as blind as ass hole Bruce was not real happy but I worked on a base where getting things to work was at least for a while more important than wasting tax payers money for crap that does work. May be your stuff is good Terry but I know GNU C is dam good. But managers like to toss money that is why Gates is so rich. He does not have to make a good operating system. Since the ones buying are not the working programers since they would use Linux.

--
http://cryptography.org/cgi-bin/crypto.cgi/Misc/scott19u.zip
for the version with a real key of voer one million bytes.
also scott16u.zip and scott4u.zip

Subject: Re: AES and patent rights
Date: Wed, 30 Sep 1998 04:39:10 GMT
From: ritter@io.com (Terry Ritter)
Message-ID: <3611b5aa.31618675@news.io.com>
References: <6urr1schf1@news.ysu.edu>
Newsgroups: sci.crypt
Lines: 74

On 29 Sep 1998 23:38:36 GMT, in <6urr1schf1@news.ysu.edu>, in
sci.crypt an096@yfn.ysu.edu (David A. Scott) wrote:

>>>[...]
>>>BTW: Do you think the development of GNU C is unfair against Borland and
>>>Microsoft?
>>
>>I guess that would first depend upon whether the government was
>>supporting GNU C, and next whether the government would be
>>recommending GNU C and even requiring it for their own use.
>
> I think most management mistakenly thinks that you have
>to pay lots of money to get something good. When I worked
>for the government we got the people at are site to use
>GNU C which produces faster running code than the SUN
>compiler. Upper management being as blind as ass hole Bruce
>was not real happy but I worked on a base where getting things
>to work was at least for a while more important than wasting
>tax payers money for crap that does work. May be your stuff
>is good Terry but I know GNU C is dam good. But managers like
>to toss money that is why Gates is so rich. He does not have
>to make a good operating system. Since the ones buying are
>not the working programers since they would use Linux.

Since I don't understand your point, I guess it is possible that you
misunderstood mine:

Do I think the development of GNU C is *unfair* to Borland and
Microsoft. Of course not.

But the question was presumably intended as an analogy to AES: The
AES competition is government funded; presumably GNU C is not. The
government may recommend AES for various uses (such as financial
transfers), and may require AES for government internal use. None of
this is like GNU C.

This really might have been a "watershed" question as in: "Since all
the right guys like GNU C, if you don't, you're not worth respecting."
I see that sort of stuff a lot, unfortunately.

Frankly, I don't know GNU C. I liked Borland from the time they
started up, and I liked a number of things about the Pascal packages,
especially including the super-fast linkage and unused code removal.
This made the development of "library" units very practical. But I
suppose the main advantage was the edit / compile / debug environment
which later became common, but seemed particularly nice in the Pascal.

Well, there is no Borland anymore, and I don't like the route taken by
Delphi environment, and my Borland C++ 4.52 environment crashes every
time I look at it. What am I going to do, get Microsoft stuff? Maybe
I'm the only guy who *never* had a wonderful experience with Microsoft
tools, but I always end up saying "damn this stupid design." Maybe
there is no real alternative now, but I'm still chugging along with
what I've got.

I sure would like to have the opportunity to get into an OS design
project and especially get rid of the old-timey load and address
fix-up stuff, and also prepare for nonvolatile main memory (which
should permit instant-on). Ideally we would innovate a new processor
architecture too, but there is only so much one can do at once. There
is a lot of stuff to do to catch up to where we could be if only our
systems were engineered to perform instead of being hacked together to
almost-always work.

As for GNU C, I don't like "copyleft," so I have stayed away from all
that stuff.

Terry Ritter ritter@io.com http://www.io.com/~ritter/
Crypto Glossary 1998-08-27: http://www.io.com/~ritter/GLOSSARY.HTM

Subject: Re: AES and patent rights
Date: Tue, 29 Sep 1998 21:13:33 -0600
From: jgfunj@EnqvbSerrGrknf.pbz (W T Shaw)
Message-ID: <jgfunj-2909982113340001@dialup163.itexas.net>
References: <6us2ol$q7e$1@news.ysu.edu>

<6ubqtt$245$1@quine.mathcs.duq.edu>
<6uqvdq$i2$1@quine.mathcs.duq.edu>
Newsgroups: sci.crypt
Lines: 17

In article <6us2ol$q7e$1@news.ysu.edu>, an096@yfn.ysu.edu (David A. Scott)
wrote:

> I am sure the NSA has already decided a long time ago who
> would get to win this farce. Only if people don't use this
> blessed crap do they have a hope of descent crypto. And I again
> say as proof just like at his chicken shit contest.
>
Current events beat the soaps. It seems that what will happen next in the
world is as much a surprise to their leadership as anything. Meanwhile,
lots of inside low-level events predictors are saying" Told you so."
--

Show me a politician who does not lie through his teeth,
and.....I'll show you one who can't find his dentures.

Decrypt with ROT13 to get correct email address.

Terry Ritter, his current address, and his top page.

Last updated: 1999-01-19

Randomness and the CRC

Path:
news.io.com!news.eden.com!mr.net!news.maxwell.syr.edu!nntp.uio.no!mn5.swip.net!news
From: bo.doemstedt@mbox200.swipnet.se (Bo Dömstedt)
Newsgroups: sci.crypt
Subject: Re: Non-deterministic random generator for the PC
Date: Thu, 30 Oct 1997 11:25:53 GMT
Organization: Protego Information AB
Lines: 56
Message-ID: <345868af.706133259@nntpserver.swip.net>
References: <19971012014201.VAA24799@ladder01.news.aol.com>
<19971014114700.HAA03888@ladder02.news.aol.com> <34438fd1.501392634@ntserv02>
<344427C8.E85EBD4A@flash.net> <344BA33C.5795@helsbreth.org>
<34576f4a.84848645@ntserv02>
Reply-To: bo.doemstedt@mbox200.swipnet.se

wayne@hoxnet.com (Wayne D. Hoxsie Jr.) wrote:
>>Much better is to take _all_ the data, even the most significant bits,
>>and distill them with a hashing function like CRC32 or MD5.
>
>
>This seems to be a decent proposal. I've tried the Von Neumann
>approach and it seemed to produce decent results (flat distribution,
>no apparent patterns), but chi-square tests showed non-random
>tendencies. I later added an option to use a 4:1 MD5 hash to the
>incoming data (512 bits in -> 128 bits out) and this method has passed
>every randomness test I've thrown at it.
>
>Its interesting to note that a 1:1 MD5 had the same results, but it
>seemed to me that if the Von Neumann method distilled the raw data
>down by about 3:1 that there must be some deterministic infomation
>getting through the MD5 hash. I settled on the 4:1 MD5 hash as a
>guess.
>
>Any comments?

Extracting the information from the output sequence is no
real problem, as evident from previous postings. Note
that "the hash method" is much superior to the
"von Neumann approach", but I would rise questions if
CRC32 or MD5 (MD4, etc.) is really suitable for this
work. For the SG100 hardware random number generator
we use a special hash-function that has been designed
especially for this task.

The main problem, in designing a hardware random number
generator and making it work, is not any of the academic
issues mentioned so far. It took about 20 times more effort
and programming making the serial port driver working,
as compared to any cryptographic-math stuff.

For the hobbyist PC-cryptographer, who want to make some
simple (and cheap!) hardware for himself, make sure to
take the issue of shielding seriously. In fact, our first
SG100 prototype, had to have no less than three levels of
filters to remove RF-signals from the power lines.

I would strongly recommend a battery powered generator
with a IR diode (light) output, all mounted in a metal box.
The battery consumption is definitively cheaper than
the approach of using excessive shielding, or the
method we use in the SG100.

The SG100 generator
http://www.protego.se/sg100_en.htm

Bo Dömstedt
Cryptographer
Protego Information AB
Malmoe,Sweden
http://www.protego.se

From: ritter@io.com (Terry Ritter)
Newsgroups: sci.crypt
Subject: Re: Non-deterministic random generator for the PC
Date: Thu, 30 Oct 1997 16:49:31 GMT
Lines: 28
Message-ID: <3458b9d4.1062007@news.io.com>
References: <19971012014201.VAA24799@ladder01.news.aol.com>
<19971014114700.HAA03888@ladder02.news.aol.com> <34438fd1.501392634@ntserv02>
<344427C8.E85EBD4A@flash.net> <344BA33C.5795@helsbreth.org>
<34576f4a.84848645@ntserv02> <345868af.706133259@nntpserver.swip.net>
NNTP-Posting-Host: as4-dialup-44.wc-aus.io.com

On Thu, 30 Oct 1997 11:25:53 GMT, in
<345868af.706133259@nntpserver.swip.net> in sci.crypt
bo.doemstedt@mbox200.swipnet.se (Bo Dömstedt) wrote:

>[...]
>Extracting the information from the output sequence is no
>real problem, as evident from previous postings. Note
>that "the hash method" is much superior to the
>"von Neumann approach", but I would rise questions if
>CRC32 or MD5 (MD4, etc.) is really suitable for this
>work. For the SG100 hardware random number generator
>we use a special hash-function that has been designed
>especially for this task.

I'd be interested in hearing more about this special hash function.
What do you see as the requirements which separate this hashing task
from others?

I have often argued that a CRC is ideal in this work. It is fast, has
a strong mathematical basis, and if it really is collecting physical
randomness, there is no need for strength.

Terry Ritter ritter@io.com http://www.io.com/~ritter/
The Crypto Glossary: http://www.io.com/~ritter/GLOSSARY.HTM

Path:
news.io.com!insync!news-feed.inet.tele.dk!news.maxwell.syr.edu!howland.erols.net!ix.netcom.com!news
From: rcktexas@ix.netcom.com (R. Knauer-AIMNET)
Newsgroups: sci.crypt
Subject: Re: Non-deterministic random generator for the PC
Date: Thu, 30 Oct 1997 20:12:43 GMT
Organization: Netcom
Lines: 11
Message-ID: <3458e9fe.2203398@nntp.ix.netcom.com>
References: <19971012014201.VAA24799@ladder01.news.aol.com>
<19971014114700.HAA03888@ladder02.news.aol.com> <34438fd1.501392634@ntserv02>
<344427C8.E85EBD4A@flash.net> <344BA33C.5795@helsbreth.org>
<34576f4a.84848645@ntserv02> <345868af.706133259@nntpserver.swip.net>
<3458b9d4.1062007@news.io.com>
Reply-To: rcktexas@ix.netcom.com

On Thu, 30 Oct 1997 16:49:31 GMT, ritter@io.com (Terry Ritter) wrote:

>I have often argued that a CRC is ideal in this work.

Would the CRC-16 suffice for cryptographically secure hashing, in your
opinion?

Bob Knauer

"Without Censorship, Things Can Get Terribly Confused In The Public Mind."
 - General William Westmoreland

From: ritter@io.com (Terry Ritter)
Newsgroups: sci.crypt
Subject: Re: Non-deterministic random generator for the PC
Date: Thu, 30 Oct 1997 22:35:32 GMT
Lines: 27
Message-ID: <34590ab6.889036@news.io.com>
References: <19971012014201.VAA24799@ladder01.news.aol.com>
<19971014114700.HAA03888@ladder02.news.aol.com> <34438fd1.501392634@ntserv02>
<344427C8.E85EBD4A@flash.net> <344BA33C.5795@helsbreth.org>
<34576f4a.84848645@ntserv02> <345868af.706133259@nntpserver.swip.net>
<3458b9d4.1062007@news.io.com> <3458e9fe.2203398@nntp.ix.netcom.com>

On Thu, 30 Oct 1997 20:12:43 GMT, in
<3458e9fe.2203398@nntp.ix.netcom.com> in sci.crypt
rcktexas@ix.netcom.com (R. Knauer-AIMNET) wrote:

>On Thu, 30 Oct 1997 16:49:31 GMT, ritter@io.com (Terry Ritter) wrote:
>
>>I have often argued that a CRC is ideal in this work.
>
>Would the CRC-16 suffice for cryptographically secure hashing, in your
>opinion?

No, no, no! All CRC's are linear operations. They essentially have
no strength at all!

But in many applications -- even in cryptography -- no strength is
needed. One example is the hashing of variable-length user key
phrases into the fixed-length arbitrary values needed to key a random
number generator. See, for example:

   http://www.io.com/~ritter/KEYSHUF.HTM

Terry Ritter ritter@io.com http://www.io.com/~ritter/
The Crypto Glossary: http://www.io.com/~ritter/GLOSSARY.HTM

From: kelsey@plnet.net
Subject: Re: Non-deterministic random generator for the PC
Date: Mon, 03 Nov 1997 11:11:39 -0600
Message-ID: <878576604.1048@dejanews.com>
Newsgroups: sci.crypt
Organization: Deja News Posting Service
Path:
news.io.com!globeset.com!news.eden.com!uunet!in1.uu.net!pressimage!news1.isdnet.net!newsfeed.nacamar.de!dispose.news.demon.net!demon!news.idt.net!ais.net!news-out.internetmci.com!newsfeed.internetmci.com!204.238.120.130!jump.net!grunt.dejanews.com!not-for-mail
Lines: 41

-----BEGIN PGP SIGNED MESSAGE-----

>Subject: Re: Non-deterministic random generator for the PC
>From: ritter@io.com (Terry Ritter)
>Date: 1997/10/30

>I have often argued that a CRC is ideal in this work. It is
>fast, has a strong mathematical basis, and if it really is
>collecting physical randomness, there is no need for
>strength.

I mostly agree with this statement. The CRC is great, if
you know that your opponent can't control your inputs. If
he can control all your inputs, he can precisely control the
input to your PRNG. If he can't see the intermediate CRC
values that you're using to collect inputs, though, he loses
control as more unknown-to-him bits appear in the input.
(That is, if he knows all but one bit of the input, then he
can guess that bit, choose the remaining inputs, and end up
with a 0.5 chance of ending up with a desired final CRC
value. If the starting state of the CRC is unknown to the
attacker, then all this kind of attacks fail.)

>Terry Ritter ritter@io.com http://www.io.com/~ritter/
>The Crypto Glossary: http://www.io.com/~ritter/GLOSSARY.HTM

   --John Kelsey, Counterpane Systems, kelsey@counterpane.com
 PGP 2.6 fingerprint = 4FE2 F421 100F BB0A 03D1 FE06 A435 7E36

-----BEGIN PGP SIGNATURE-----
Version: 2.6.2

iQCVAwUBNF1aP0Hx57Ag8goBAQEKkQQA1vUvE37vJiEjJet65JS2jOC7PVirPRQ2
FrXcj5dlkLQZiNOEu7REb1/bM8ZGzbVtFaVAPw/R71ACKOJMHI0nSIJk2PHDu+oW
tUggFgZDhuytJtbG4dweQm2NC7LsNZLOTe/1sRep2Nobeur6xAcFN5v9VrbWzAtM
ox4754cavL4=
=/7J2
-----END PGP SIGNATURE-----

--------------------==== Posted via Deja News ====-----------------------
      http://www.dejanews.com/   Search, Read, Post to Usenet

---

Terry Ritter, his current address, and his top page.

Last updated: 1998-01-16

Randomness Tests; Blum, Blum & Shub

General References on Testing RNG's

1. Beker, H. and F. Piper. 1982. Cipher Systems. Wiley. 169-174.
2. Beker, H. and F. Piper. 1985. Secure Speech Communications. Academic Press. 104-109.
3. Carroll, J. 1989. The binary derivative test: noise filter, crypto aid, and random-number seed selector. *Simulation*. 53(3): 129-135.
4. Feldman, F. 1990. Fast Spectral Tests for Measuring Nonrandomness and the DES. *IEEE Transactions on Software Engineering*. 16(3): 261-267. (Also in *Advances in Cryptology -- CRYPTO '87*.)
5. Knuth, D. 1981. The Art of Computer Programming. Vol. 2. Seminumerical Algorithms, 2nd Ed. Addison-Wesley. 38-110.
6. L'Ecuyer, P. 1992. Testing Random Number Generators. *Proceedings of the 1992 Winter Simulation Conference*. 305-313.
7. Marsaglia, G. and A. Zaman. 1993. Monkey Tests for Random Number Generators. *Computers & Mathematics with Applications*. 26(9): 1-10.
8. Marsaglia, G. 1985. A Current View of Random Number Generators. *Proceedings of the Sixteenth Symposium on the Interface between Computer Science and Statistics*. 3-10.
9. Marsaglia, G. 1985. Note on a Proposed Test for Random Number Generators. *IEEE Transactions on Computers*. c-34(8): 756-758.
10. Maurer, U. 1990. A Universal Statistical Test for Random Bit Generators. *Advances in Cryptology -- CRYPTO '90*. 409-420.
11. Mund, S. 1991. Ziv-Lempel Complexity for Periodic Sequences and its Cryptographic Application. *Advances in Cryptology -- EUROCRYPT '91*. 114-126.
12. Newman, E. 1951. Computational Methods Useful in Analyzing Series of Binary Data. *American Journal of Psychology*. 64: 252-262.
13. Richards, T. 1989. Graphical Representation of Pseudorandom Sequences. *Computers & Graphics*. 13(2): 261-262.

Randomness Tests

Everybody has a favorite.

- 1993-02-25 Ian Collier: (some comments on RNG implementation)
- 1993-12-09 George Fuellen: The really clean definition of randomness is an asymptotic one . . . . There are no "random" strings, but there can be algorithms that have a "random" output.
- 1994-02-02 Herman Rubin: If you want something good enough for Monte Carlo, it is impossible to test for it, because to test, say, that the bits are accurate to .001, a sample of about 10^7 will be needed, and for any complicated Monte Carlo, this is nowhere near good enough. Mere testing is not good enough.
- 1994-10-18 Charles Stevens: I have produced some programs to analyise the "randomness" properties of a binary sequence.

Testing Hardware RNG's

- 1994-08-04 Ross Anderson: If you take a number of 32-bit samples, then you should start finding collisions (values sampled twice) after you have drawn about D = 2^16 samples, values sampled three times after about T = 2^22 samples (actually 3N^{2/3}), and so on.
- 1994-08-04 Terry Ritter: For any particular population, the number of doubles in a trial of a particular size follows a distribution which is Poisson-like. Thus, any single trial can have an extremely wide range of results.
- 1994-08-06 Terry Ritter: (Subject: Re: multi-sided dice for RNGs)

Maurer's Test

Is it the ultimate window on reality?

- 1993-12-03 Peter K. Boucher: Here's a little ditty that passes Maurer and is not random at all.
- 1993-12-07 Peter K. Boucher: I modified the Maurer test poster earlier by Dimitri Vulis to get rid of the Q input. Q is the number of ``random" values to skip so that, hopefully, at least one of each value has been seen before the actual testing begins.
- 1994-05-14 Donald T. Davis: maurer points out that in calculating H = -1 * sum(p*log2p), you can replace p, a bitstring's probability of appearance, with the interarrival time between repetitions of the bitstring.
- 1994-05-16 Mike Scott: While not good enough to test for Crypto strength RNGs . . . any such generator should pass this test
- 1994-06-14 Donald T. Davis: please realize that entropy estimation isn't any more powerful at detecting structure than standard statistical measures are
- 1994-10-31 Steve Allen: How universal is Ueli Maurer's Universal Statistical Test For Random Bit Generators?
- 1994-11-01 Mark Johnson: Regrettably, Maurer's test does not reject the infamous RANDU algorithm.
- 1994-11-01 Donald T. Davis: an entropy estimate is necessarily a very imperfect measure
- 1994-12-10 Steve Allen: See Ueli Maurer's test: He claims, as I understand it, that this test reveals *any* weakness in a RNG due to memory
- 1994-12-11 Terry Ritter: (quoting Nick Maclaren) "It is important to note that universal tests have been known to statisticians for half a century, but are little used because they tend to be very weak."

I sure didn't do a very good job formatting that last message, but as you can see it was basically just the earlier messages, plus one other reference. I included very few of my own comments. But apparently someone didn't like the implications, and sent that message to Ueli Maurer, perhaps with some goading comments. I consequently had a brief, sharp exchange with Ueli in private e-mail. Now, Ueli Maurer is a well-known, respected and very accomplished researcher in cryptography, and I have no "hidden agenda" either to damage his reputation or to improve my own at his expense (I am not an academic and need play no petty academic games). But in this particular case, the Title and Abstract of the paper can be *extremely* misleading to those not deeply involved with these particular issues.

1. First of all, Maurer's paper is intended to apply to physically-random generators, and specifically disclaims the ability to test "software" RNG's. But randomness tests apply to *sequences*, not *generators*. How does the test know where a sequence comes from?
2. Suppose we do test a physically-random generator. Also suppose that the fault in that generator is that it is a software-like state-machine. (Generators based on a multitude of crystal oscillators might be a good example.) It is not satisfactory for a test to pass such a sequence because the physically-random generator internally behaves in a way to which the test does not apply.

   Indeed, I would argue that even physically-random generators *should* be considered to be state-machines with essentially infinite state. (For one thing, this allows us to avoid "mystical" answers about the ultimate source of ever-increasing entropy.) This would mean that both "physical" and "software" RNG's could be analyzed on the same continuium.
3. The Maurer test supposedly delivers the "entropy" of a sequence.

   Q: What is the entropy of a 32-bit RNG?
   A: 32 bits.

   If the entropy estimator does not get this, or if it takes impractically long to do so, it is not a very good estimator, or it is measuring some other definition of "entropy." But Mark Johnson (above) tells us that this test does not reject a well-known problem generator with minimal internal state.

   This is how we check randomness tests: We generate sequences with known characteristics and then measure them. Tests which fail are just failed tests. Failed tests are precisely what we do **not** use to analyze sequences with unknown characteristics. And new physical RNG designs or implementations are often rife with potential problems.

I consider the Maurer test to be another statistical test, useful for investigating it's own particular view of randomness. I do not, however, consider it to be a complete report on any possible problem in the generator, even in a physically-random generator.

Blum, Blum & Shub

Basically, the function x^2 Mod N used iteratively. The attraction is a proof of "unpredictability," which seems like it would solve "the" stream-cipher problem.

There are both theoretic and practical problems: The proof is asymptotic, and therefore provides no guidance as to the size of N needed for "unpredictability." (Presumably, it requires RSA-size values.) Practical problems include the need for other restrictions on P, Q (N = PQ), *finding* P,Q, and selecting x0.

- 1994-06-17 Robert I. Eachus: AFAIK the only algorithmic RNG that can be considered secure and has reasonable performance is Blum, Blum, and Shub.
- 1994-06-18 Terry Ritter: The primary BB&S requirement for N = P * Q is that P and Q each be primes congruent to 3 mod 4. This is exceedingly easy. But to guarantee a particular cycle length (Section 8, Theorem 6), there are two more Conditions [1:378]. Condition 1 defines that a prime is "special" if P = 2*P1 + 1, and P1 = 2*P2 + 1, where P1 and P2 are odd primes. Both P and Q are required to be "special." Finding a valid special primes P and Q is not easy.
- 1995-03-08 Peter Kwangjun Suk: How about using the secure Blum-Blum-Shub quadratic residue generator with a 512 bit modulus? We could run a fast 64-bit block cipher in OFB mode, and occasionally flush the shift register and insert 63 bits from seven iterations of the BBS generator.
- 1995-03-13 Peter Kwangjun Suk: From what I've read, BBS is secure, both to the left and to the right, so long as you only use a few (~lg n) bits of each X_(j).
- 1995-03-15 Don Beaver: there may be very simple and very "efficient" *polynomial-time* attacks that predict BBS for n<1000000. Alexi/Chor/Goldreich/Schnorr would not be contradicted in the least. Such attacks would merely _eventually_ fail. *How big* n has to be is not a part of the theoretical rating. There is no theoretical support to deny the existence of "efficient," polynomial-time attacks on BBS for n<1000000.
- 1995-03-19 Robert I. Eachus: Huh? The original Blum, Blum and Shub paper proved that at least one secure bit could be generated each iteration.
- 1995-03-20 Peter Kwangjun Suk: Moderation on this group seems to have produced a false positive in this case.
- 1995-03-21 Don Beaver: These results are only *asymptotic* results. Their implications have been strongly misunderstood. They say, essentially, that any (polytime) algorithm that tries to predict a BBS generator using a window of 1 bit [or lg2(N) bits, if you prefer] will have virtually no advantage, *for large enough N*. They say nothing about the success or failure of polytime algorithms on N that have fewer than a few thousand bits.
- 1995-03-22 Don Beaver: (reply to Bob)
- 1995-03-26 Don Beaver: (reply to Peter) complexity-theoretic results can't be used to say that lg(512)-bit windows of a BBS generator are secure.
- 1995-10-06 Robert I. Eachus: The Blum, Blum and Shub algorithm is at least as difficult to predict as it is to factor the modulus.
- 1995-10-07 Nick Maclaren: PLEASE remember that this paper is SERIOUSLY flawed, and the technique is NOTHING LIKE as good as it is claimed to be.
- 1995-10-08 Herman Rubin: the generator is far too slow to be of much use, unless one needs fantastically good pseudo-random numbers, and can afford the cost . . . A Tausworthe generator like x[n] = x[n-460] + x[n-607] has period 2^(s-1)*(2^607 -1), where s is the word length; this is in integer arithmetic. This class of procedures are now known to have drawbacks.
- 1995-10-09 Arthur Chance Could you explain that last sentence?
- 1995-10-09 Herman Rubin: the bad example was such a generator with a largest lag of 1279. In an Ising model for which the results were known, simulation gave wrong answers.
- 1995-10-11 Robert I. Eachus: The second bag of tricks and the one that makes it all practical is to use the Chinese Remainder Theorem to allow you to do all of the computations with machine arithmetic operations instead of multiprecision.
- 1995-10-14 Nick Maclaren: (comments on Tausworthe problems)

Unbiased Range Reduction for RNG's

Suppose we have a RNG which produces values with a uniform probability of selecting any particular value in its range. Then suppose we want a lesser range. There are several wrong ways to do this.

- 1994-02-24 Carl Ellison: For example, if ranno() were to return a number in the range 0..14 and x were 10, then (ranno() % x) would produce an element in [0..4] twice as often as an element in [5..9]. So, the distribution is not uniform.
- 1994-02-24 John Kelsey: Suppose you had a random number generator that always put out a uniformly distributed number between 0 and 10. Now, imagine trying to get evenly distributed decimal digits by taking the output of that function modulo 10. You'd get twice as many 0s as any other number, right?
- 1994-02-25 Eli Brandt: Related problem: you often see code like `if (!(rand()%1000)) { ... }` with the intent that the block be run with probability .001, which it won't be.
- 1994-02-25 Carl Ellison: (correction)
- 1995-06-01 Adam Back: Here's a method which I think solves this problem.
- 1995-06-02 Peter da Silva: Or what I usually do:
- 1995-06-05 Felix Schroeter: (comments)

Terry Ritter, his current address, and his top page.

Last updated: 1995-12-28

Really Random Topics

Really Random Generators

Electronic hardware for generating really-random values.

- 1990-??-?? John Rogers: Use a 5v op-amp to amplify base-emitter breakdown noise
- 1990-??-?? David Battle: Use a low-frequency oscillator to sample a high-frequency oscillator and clock the bits into a shift-register
- 1990-10-04 Steven Bellovin: The two-oscillator scheme is similar to that used in the AT&T random number chip
- 1990-??-?? Kevin Quitt: Use an inverter as a high-frequency oscillator to clock a counter, then read the counter at random.
- 1990-??-?? Ted Dunning: In the two-oscillator scheme, inject noise into the low-frequency oscillator signal.
- 1992-03-18 William Unruh: Instruments used to detect randomness may introduce biases and correlations.
- 1992-06-18 Bill Squire: Use highpass filtered diode noise to toggle a flip-flop, and sample at random.
- 1992-06-19 Phil Karn: Why *not* use a PC sound card to digitize FM receiver noise?
- 1992-06-19 Paul Rubin: Paul responds to Phil: It's hard to get good randomness from schemes like this.
- 1992-10-22 Peter Gutmann: A circuit using a zener diode amplified by an op-amp.
- 1992-10-22 Eric Backus: Eric comments on Peter's circuit.
- 1992-10-24 Ross Anderson: Ross responds to Eric: You have to be careful here.
- 1994-02-02 Wray Kephart: Wray suggests using a commercial randomizer chip.
- 1994-02-02 Herman Rubin: Comments on testing really-random RNG's.

Improving Randomness

Physical randomness is processed before use.

- 1990-??-?? Mark Johnson: Santha-Vazirani, or the effect of XORing multiple "semi-random" sources.
- 1990-??-?? Mark Johnson: More Santha-Vazirani.
- 1990-06-14 David Lewis: A different algorithm, supposedly more accurate and more efficient.
- 1990-??-?? Benny Pinkas: The Lewis algorithm is similar to work by Blum.
- 1991-??-?? Steve Tate: There is "bias," and then there is "bias."
- 1991-??-?? Herman Rubin: More comments on bias.
- 1991-09-25 Mark Johnson: Get an even bit distribution by XORing random bits with a square wave.
- 1991-09-26 Herman Rubin: The general idea is old.
- 1992-03-18 Topher Cooper: Use DES to hash (and protect) random bits, and use a subset of them.

Essential Randomness

Is there any?

- 1990-??-?? Albert Boulanger: The randomness in Quantum Mechanics (QM) is not necessarily axiomatic.
- 1991-08-04 Paul Budnik: There is no basis for the belief that randomness is fundamental to quantum mechanics.
- 1991-08-05 Hugh Miller: Hugh quotes Penrose.
- 1991-08-06 Doug Gwin: The known phenomena of quantum physics are incompatible with local determinism.
- 1991-08-06 Paul Budnik: Paul responds to Hugh.
- 1991-08-06 Paul Budnik: Local determinism in general is not equivalent to the determinism advocated by Einstein in EPR.
- 1991-08-11 Albert Boulanger: The axiomatic development is deliberately silent concerning any requirements that the measurable functions be non-determinate.
- 1991-08-13 Lee Campbell: Nobody has come up with a working hidden variable theory.
- 1991-08-19 Paul Budnik: Local quantum mechanics is consistent with standard quantum mechanics on all known experimental results. It differs from standard quantum mechanics because it does not violate locality.
- 1991-08-14 Doug Gwin: The quantum "randomness" is not merely ASSUMED, it has been DEMONSTRATED and thus is a required feature of any such theory.
- 1991-08-26 Paul Budnik: It is false to claim that quantum randomness has been conclusively demonstrated or proved and it is unscientific to assume that there cannot be a more complete theory without such evidence or proof.

Terry Ritter, his current address, and his top page.

Last updated: 1995-10-31

Random Numbers in C

A *Ciphers By Ritter* Page

Random number generators (*pseudo*-random, of course) in C; mostly fast one-liners for "inline" use (to eliminate call / return overhead). If you just want to *use* these RNG's, it might be a good idea to start at the last message.

Contents

- 1999-01-12 George Marsaglia: "This posting ends with 17 lines of C code that provide eight different in-line random number generators, six for random 32-bit integers and two for uniform reals in (0,1) and (-1,1)."
- 1999-01-12 Charles Bond: "Do you know if any theoretical work has been done since Knuth's book to justify SWB?"
- 1999-01-12 George Marsaglia: "It hasn't been very many years since I invented the subtract-with-borrow method, and developed theory for establishing the periods."
- 1999-01-12 Charles Bond: "I did not mean to imply that Knuth's subtractive generator was *the same* as your subtract with borrow, only that it was *similar* (high speed, no multiplications)."
- 1999-01-12 Mike Oliver: "Could you give us a pointer to information about why these RNGs [lagged Fibonacci generators using xor] are unsatisfactory and what sort of test they tend to fail?"
- 1999-01-13 Eric Backus: "I have a small problem with the definition of LFIB4 and SWB. In an attempt to make these a single line of C code, they both use "++c" in the same expression as they use "c". A C compiler is free to rearrange the order in which it calculates the intermediate terms of these expressions, so the expressions can produce different results depending on the compiler."
- 1999-01-15 George Marsaglia: "The generators MWC, KISS, LFIB4 and SWB seem to pass all tests. By themselves, CONG and SHR3 do not, but using CONG+SHR3 provides one of the fastest combinations that satisfy the DIEHARD battery of tests."
- 1999-01-15 Dmitri Zaykin: "Shouldn't these be...."
- 1999-01-01 Radford Neal: "This doesn't work either."
- 1999-01-15 Jeff Stout: "The preprocessor is just doing a stupid text substituation, its the C compiler that is ambigous about the interpretation."
- 1999-01-01 Radford Neal: "...although one can indeed create lots of junk using the C pre-processor, one cannot, in general, use it to create in-line procedures."
- 1999-01-16 Dmitri Zaykin: "It is probably true that the posted random number generators are better implemented as regular functions."
- 1999-01-16 Dmitri Zaykin: "These should work...."
- 1999-01-17 Ramin Sina: "Could someone please post a short example code on how to use these in practice."
- 1999-01-18 Ed Hook: "No -- 'SHR3' and 'LFIB4' are OK, but your version of 'SWB' still invokes the dreaded undefined behavior."
- 1999-01-19 Orjan Johansen: "Surely in an expression of the form y=..., the evaluation of ... is guaranteed to happen before the assignment."
- 1999-01-19 Horst Kraemer: "Sorry. No sequence point is needed between reading an writing to an lvalue."
- 1999-01-17 Herman Rubin: "I think it should be clarified, and probably written out in some more detail. But the procedure call overhead would be comparable to the computing cost...."
- 1999-01-17 Duncan Murdoch: "I don't know if that's true or not, but I don't think it is really something to worry about."
- 1999-01-18 Herman Rubin: "It certainly is. The cost of one procedure call could far exceed the cost of a uniform, or even many nonuniform, random variables."
- 1999-01-01 Radford Neal: "Even on a CISC machine like a VAX, I doubt the C procedure call overhead would be more that twice the time for the actual work, and on modern RISC machines the ratio will tend to be less."
- 1999-01-19 Herman Rubin: "The code discussed was not in making a call for an array, but for a single step in the generation procedure."
- 1999-01-19 Duncan Murdoch: "I'm not sure what you have in mind."
- 1999-01-19 Herman Rubin: "Can we afford range checking?" "Basic random tools should be as a bit stream, not as floating numbers, for many reasons."
- 1999-01-20 Duncan Murdoch: "...all languages that I know (including C) provide ways of forcing the order of evaluation: just put the things that need to be done first in an earlier statement."
- 1999-01-17 David W. Noon: "I don't mind translating all this stuff into assembler, but anybody not using an Intel 80486 or better running a 32-bit version of OS/2 will not be able to use my code."
- 1999-01-18 Dmitri Zaykin: "Another option is to re-write macros as C++ member functions."
- 1999-01-18 David W. Noon: "The problem with C++ is that VMT calling mechanisms are almost invariably slower than the calling mechanisms used by C, PL/I, FORTRAN, Pascal, etc."
- 1999-01-18 Dmitri Zaykin: "Well, in this particular case all that does not apply, since there are no virtual functions in the code."
- 1999-01-19 Duncan Murdoch: "...but just as with any other aspect of the program, speed isn't as important as correctness."
- 1999-01-20 Dmitri Zaykin: "To check if C-macros for these random number generators do indeed always produce faster, and maybe 'less bloated' code than inlined C++ member functions, I did a little experiment...."
- 1999-01-21 Dmitri Zaykin:
- 1999-01-21 John E. Davis: "Using your code (t.cc and t.c), my conclusion is the opposite...."
- 1999-01-21 Dmitri Zaykin: "I used egcs-1.1.1 compiler. It is is an improvement over gcc-2.8.1."
- 1999-01-18 Mike McCarty:
- 1999-01-18 Herman Rubin: "Decent random number code cannot be portable."
- 1999-01-19 David W. Noon: "I meant portable across software platforms, not hardware."
- 1999-01-19 Herman Rubin: "It is by no means clear that HLL generated code will do this, as the machine instructions involved are different." "What was meant was the distance to the next one in a stream of random bits. In other words, generate a geometric (.5) random variable using only the number of bits required by information."
- 1999-01-19 David W. Noon: "C++ just isn't ideally suited to the task under current implementations of the language."
- 1999-01-20 Dmitri Zaykin: "Templates can be an alternative to C-macros in terms of genericity, not inlining."
- 1999-01-18 Mike McCarty: "There are no 'idiot-proof languages'."
- 1999-01-18 Herman Rubin: "A language has a built in inefficiency in every situation in which the optimal use of machine instructions cannot be addressed in that language."
- 1999-01-21 Mike McCarty: "The checking I had in mind costs *nothing* during the execution of the program."
- 1999-01-16 Dmitri Zaykin: "I see one more problem with the code."
- 1999-01-19 qscgz@my-dejanews.com: "DIEHARD-tests fail on...."
- 1999-01-20 George Marsaglia: "My offer of RNG's for C was an invitation to dance; I did not expect the Tarantella." "Numerous responses have led to improvements; the result is the listing below...."

```
Subject: Random numbers in C: Some suggestions.
Date: Tue, 12 Jan 1999 09:37:37 -0500
From: George Marsaglia &LT;geo@stat.fsu.edu>
Message-ID: &LT;369B5E30.65A55FD1@stat.fsu.edu>
Newsgroups: sci.stat.math,sci.math,sci.math.num-analysis,sci.crypt,sci.physics
Lines: 243
```

This posting ends with 17 lines of
C code that provide eight different
in-line random number generators, six for
random 32-bit integers and two for uniform
reals in (0,1) and (-1,1).
Comments are interspersed with that
code. Various combinations of the six in-line
integer generators may put in C expressions to
provide a wide variety of very fast, long period,
well-tested RNG's. I invite comments, feedback,
verifications and timings.

First, there is narrative giving background
for this posting; you may want to skip it.

Narrative:

Having had experience in producing and
testing for randomness in computers,
I am frequently asked to suggest good
random number generators (RNG's), test
RNG's, or comment on existing RNG's. Many
recent queries have been in two areas:
(1) requests for implementations in C and
(2) comments on generators with immense periods,
particularly the Mersenne Twister.

This posting is mainly for category (1),
for which I suggest a set of C implementations
of RNG's I have developed. C implementations
of my DIEHARD battery of tests will be
discussed elsewhere, and Narasimhan's GUI
version is expected to be released soon.

For (2), I merely repeat what I have said
in response to various queries: the Mersenne
Twister looks good, but it seems to be essentially
a lagged Fibonacci RNG using the exclusive-or
(xor) operation, and experience has shown that
lagged Fibonacci generators using xor provide
unsatisfactory 'randomness' unless the lags are
very long, and even for those with very long lags,
(and even for those using + or - rather than xor),
many people (I among them) are inclined to be
cautious about sequences based on such a simple
operation as: each new integer is the xor, (or sum,
or difference), of two earlier ones. To be sure,
the resulting integers can be "twisted", but not,
I think, as simply or as safely as combining, say
by addition, with members of a sequence from a
(shorter period) generator that has itself passed
extensive tests of randomness.

I also reply that it does not take an immense
program (as, for example, in posted listings
of Twister) to produce a more satisfactory RNG
with an immense period, and give this example,
on which I will expand below: Inclusion of

```
#define SWB ( t[c+237]=(x=t[c+15])-(y=t[++c]+(x&LT;y)) )
```

together with suitably initialized seeds in

```
static unsigned long x,y,t[256]; unsigned char c;
```

will allow you to put the string SWB in any C
expression and it will provide, in about 100 nanosecs,
a 32-bit random integer with period 2^7578. (Here
and below, ^ means exponent, except in C expressions,
where it means xor (exclusive-or).

Now for the (2) part, in which I suggest a number
of C implementations and invite comment and feedback.
Most of these were previously developed and tested
via Fortran versions. I list eight RNG's, each of
them by means of C's powerful #define device. This
provides fast, compact implementation, allows
insertion of the required random variable directly
into an expression, and, finally, provides a good
selection of RNG's for use individually or in
combination. The latter makes it possible to
further confirm what empirical results suggest:
combining two or more RNG's provides better,
(or no worse) randomness, and for encryption enthusiasts:
combination generators are harder to "crack".

For those wishing to try these eight RNG's:

At the top of your C program, include these
definitions and the static variables that follow.
Everything past this line is either C code or comment.

```
#define UL unsigned long
#define znew  ((z=36969*(z&65535)+(z>>16))&LT;&LT;16)
#define wnew  ((w=18000*(w&65535)+(w>>16))&65535)
#define MWC   (znew+wnew)
#define SHR3  (jsr=(jsr=(jsr=jsr^(jsr&LT;&LT;17))^(jsr>>13))^(jsr&LT;&LT;5))
#define CONG  (jcong=69069*jcong+1234567)
#define KISS  ((MWC^CONG)+SHR3)
#define LFIB4 (t[c]=t[c]+t[c+58]+t[c+119]+t[++c+178])
#define SWB   (t[c+237]=(x=t[c+15])-(y=t[++c]+(x&LT;y)))
#define UNI   (KISS*2.328306e-10)
#define VNI   ((long) KISS)*4.656613e-10
/*  Global static variables: */
 static UL z=362436069, w=521288629, jsr=123456789, jcong=380116160;
 static UL t[256];
 static UL x=0,y=0; static unsigned char c=0;

/* Random seeds must be used to reset z,w,jsr,jcong and
the table t[256]  Here is an example procedure, using KISS: */

 void settable(UL i1,UL i2,UL i3,UL i4)
 { int i; z=i1;w=i2,jsr=i3; jcong=i4;
 for(i=0;i&LT;256;i++)  t[i]=KISS;        }

/*  End of C code;  Only comments follow. Stick the above
   17 lines in your simulation programs, initialize the table,
   and have a variety of promising RNG's at your disposal.  */

/* You may want use more complicated names for the
   above simple 1-letter variable names: z,w,x,y,t,c,
   to avoid clashing with favorites in your code.    */

/* Any one of KISS, MWC, LFIB4, SWB, SHR3, or CONG
   can be used in an expression to provide a random
   32-bit integer, and UNI in an expression will
   provide a random uniform in (01), or VNI in (-1,1).
   For example, for int i, float v; i=(MWC>>24); will
   provide a random byte, while v=4.+3.*UNI; will
   provide a uniform v in the interval 4. to 7.
   For the super cautious, (KISS+SWB) in an expression
   would provide a random 32-bit integer from
   a sequence with period > 2^7700, and would only
   add some 300 nanoseconds to the computing
   time for that expression.                         */

/* The KISS generator, (Keep It Simple Stupid), is
   designed to combine the two multiply-with-carry
   generators in MWC with the 3-shift register SHR3
   and the congruential generator CONG, using
   addition and exclusive-or. Period about 2^123. It
   is one of my favorite generators.   */

/* The  MWC generator concatenates two 16-bit
  multiply-with-carry generators, x(n)=36969x(n-1)+carry,
  y(n)=18000y(n-1)+carry  mod 2^16, has  period  about
  2^60 and seems to pass all tests of randomness. A favorite
  stand-alone generator---faster than KISS, which contains it.*/

/* SHR3 is a 3-shift-register generator with
   period 2^32-1. It uses
   y(n)=y(n-1)(I+L^17)(I+R^13)(I+L^5), with the
   y's viewed as binary vectors, L the 32x32
   binary matrix that shifts a vector left 1, and
   R its transpose.  SHR3 seems to pass all except
   the binary rank test, since 32 successive
   values, as binary vectors, must be linearly
   independent, while 32 successive truly random
   32-bit integers, viewed as binary vectors, will
   be linearly independent only about 29% of the time.   */

/* CONG is a congruential generator with the
   widely used 69069 as multiplier:
   x(n)=69069x(n-1)+1234567.  It has period 2^32.
   The leading half of its 32 bits seem to pass
   all tests, but bits in the last half are too
   regular.                                  */

/* LFIB4 is an extension of the class that I have
   previously defined as  lagged Fibonacci
   generators: x(n)=x(n-r) op x(n-s), with the x's
   in a finite set over which there is a binary
   operation op, such as +,- on integers mod 2^32,
   * on odd such integers, exclusive-or (xor) on
   binary vectors. Except for those using
   multiplication, lagged Fibonacci generators
   fail various tests of randomness, unless the
   lags are very long.  To see if more than two
   lags would serve to overcome the problems of 2-
   lag generators using +,- or xor, I have
   developed the 4-lag generator LFIB4:
   x(n)=x(n-256)+x(n-179)+x(n-119)+x(n-55) mod 2^32.
   Its period is 2^31*(2^256-1), about 2^287, and
   it seems to pass all tests---in particular,
   those of the kind for which 2-lag generators
   using +,-,xor seem to fail.  For even more
   confidence in its suitability,  LFIB4 can be
```

```
   combined with KISS, with a resulting period of
   about 2^410: just use (KISS+LFIB4) in any C
   expression.                                */

/* SWB is a subtract-with-borrow generator that I
   developed to give a simple method for producing
   extremely long periods:
   x(n)=x(n-222)-x(n-237)-borrow mod 2^32.
   The 'borrow' is 0 unless set to 1 if computing
   x(n-1) caused overflow in 32-bit integer
   arithmetic. This generator has a very long
   period, 2^7098(2^480-1), about 2^7578. It seems
   to pass all tests of randomness, but,
   suspicious of a generator so simple and fast
   (62 nanosecs at 300MHz), I would suggest
   combining SWB with KISS, MWC, SHR3, or CONG. */

/* Finally, because many simulations call for
   uniform random variables in 0&LT;v&LT;1 or -1&LT;v&LT;1, I
   use #define statements that permit inclusion of
   such variates directly in expressions:  using
   UNI will provide a uniform random real (float)
   in (0,1), while VNI will provide one in (-1,1).  */

/* All of these: MWC, SHR3, CONG, KISS, LFIB4,
   SWB, UNI and VNI, permit direct insertion of
   the desired random quantity into an expression,
   avoiding the time and space costs of a function
   call. I call these in-line-define functions.
   To use them, static variables z,w,jsr and
   jcong should be assigned seed values other than
   their initial values.  If LFIB4 or SWB are
   used, the static table t[256] must be
   initialized.  A sample procedure follows. */

/* A note on timing:  It is difficult to provide
   exact time costs for inclusion of one of these
   in-line-define functions in an expression.
   Times may differ widely for different
   compilers, as the C operations may be deeply
   nested and tricky. I suggest these rough
   comparisons, based on averaging ten runs of a
   routine that is essentially a long loop:
   for(i=1;i&LT;10000000;i++) L=KISS; then with KISS
   replaced with SHR3, CONG,... or KISS+SWB, etc.
   The times on my home PC, a Pentium 300MHz, in
   nanoseconds: LFIB4=64; CONG=90; SWB=100;
   SHR3=110; KISS=209; KISS+LFIB4=252; KISS+SWB=310.     */
```

Subject: Re: Random numbers in C: Some suggestions.
Date: Tue, 12 Jan 1999 07:13:47 -0800
From: Charles Bond <cbond@ix.netcom.com>
Message-ID: <369B66AB.6CDED9F8@ix.netcom.com>
References: <369B5E30.65A55FD1@stat.fsu.edu>
Newsgroups: sci.stat.math,sci.math,sci.math.num-analysis,sci.crypt,sci.physics
Lines: 253

Thanks for the post. I want to comment on the SWB routine. I've been
using
a similar routine in high speed simulations for years. Small departures
from
statistically correct randomness are not a problem for my application,
but
speed is. I believe Knuth briefly discussed the method with guarded
approval -- constrained by the concern that there was no real theory
behind it. Do you know if any theoretical work has been done since
Knuth's
book to justify SWB?

George Marsaglia wrote:

> This posting ends with 17 lines of
> C code that provide eight different
> in-line random number generators, six for
> random 32-bit integers and two for uniform
> reals in (0,1) and (-1,1).
> Comments are interspersed with that
> code. Various combinations of the six in-line
> integer generators may put in C expressions to
> provide a wide variety of very fast, long period,
> well-tested RNG's. I invite comments, feedback,
> verifications and timings.
>
> First, there is narrative giving background
> for this posting; you may want to skip it.
>
> Narrative:

```
>
> Having had experience in producing and
> testing for randomness in computers,
> I am frequently asked to suggest good
> random number generators (RNG's), test
> RNG's, or comment on existing RNG's.  Many
> recent queries have been in two areas:
> (1) requests for implementations in C and
> (2) comments on generators with immense periods,
> particularly the Mersenne Twister.
>
> This posting is mainly for category (1),
> for which I suggest a set of C implementations
> of RNG's I have developed.  C implementations
> of my DIEHARD battery of tests will be
> discussed elsewhere, and Narasimhan's GUI
> version is expected to be released soon.
>
> For (2), I merely repeat what I have said
> in response to various queries: the Mersenne
> Twister looks good, but it seems to be essentially
> a lagged Fibonacci RNG using the exclusive-or
> (xor) operation, and experience has shown that
> lagged Fibonacci generators using xor provide
> unsatisfactory 'randomness' unless the lags are
> very long, and even for those with very long lags,
> (and even for those using + or - rather than xor),
> many people (I among them) are inclined to be
> cautious about sequences based on such a simple
> operation as: each new integer is the xor, (or sum,
> or difference), of two earlier ones.  To be sure,
> the resulting integers can be "twisted", but not,
> I think, as simply or as safely as combining, say
> by addition, with members of a sequence from a
> (shorter period) generator that has itself passed
> extensive tests of randomness.
>
> I also reply that it does not take an immense
> program (as, for example, in posted listings
> of Twister) to produce a more satisfactory RNG
> with an immense period, and give this example,
> on which I will expand below: Inclusion of
>
> #define SWB ( t[c+237]=(x=t[c+15])-(y=t[++c]+(x&LT;y)) )
>
> together with suitably initialized seeds in
>
> static unsigned long x,y,t[256]; unsigned char c;
>
> will allow you to put the string SWB in any C
> expression and it will provide, in about 100 nanosecs,
> a 32-bit random integer with period  2^7578. (Here
> and below, ^ means exponent, except in C expressions,
> where it means xor (exclusive-or).
>
> Now for the (2) part, in which I suggest a number
> of C implementations and invite comment and feedback.
> Most of these were previously developed and tested
> via Fortran versions.  I list eight RNG's, each of
> them by means of C's powerful #define device. This
> provides  fast, compact implementation, allows
> insertion of the required random variable directly
> into an expression, and, finally, provides a good
> selection of RNG's for use individually or in
> combination.  The latter makes it possible to
> further confirm what empirical results suggest:
> combining two or more RNG's provides better,
> (or no worse) randomness, and for encryption enthusiasts:
> combination generators are harder to "crack".
>
> For those wishing to try these eight RNG's:
>
> At the top of your C program, include these
> definitions and the static variables that follow.
> Everything past this line is either C code or comment.
> --------------------------------------------------
>
> #define UL unsigned long
> #define znew  ((z=36969*(z&65535)+(z>>16))&LT;&LT;16)
> #define wnew  ((w=18000*(w&65535)+(w>>16))&65535)
> #define MWC   (znew+wnew)
> #define SHR3  (jsr=(jsr=(jsr=jsr^(jsr&LT;&LT;17))^(jsr>>13))^(jsr&LT;&LT;5))
> #define CONG  (jcong=69069*jcong+1234567)
> #define KISS  ((MWC^CONG)+SHR3)
> #define LFIB4 (t[c]=t[c]+t[c+58]+t[c+119]+t[++c+178])
> #define SWB   (t[c+237]=(x=t[c+15])-(y=t[++c]+(x&LT;y)))
> #define UNI   (KISS*2.328306e-10)
> #define VNI   ((long) KISS)*4.656613e-10
> /*  Global static variables: */
>  static UL z=362436069, w=521288629, jsr=123456789, jcong=380116160;
>  static UL t[256];
>  static UL x=0,y=0; static unsigned char c=0;
>
> /* Random seeds must be used to reset z,w,jsr,jcong and
> the table t[256]  Here is an example procedure, using KISS: */
```

```
>
>  void settable(UL i1,UL i2,UL i3,UL i4)
>  { int i; z=i1;w=i2,jsr=i3; jcong=i4;
>  for(i=0;i&LT;256;i++)  t[i]=KISS;        }
>
> /*  End of C code;  Only comments follow. Stick the above
>    17 lines in your simulation programs, initialize the table,
>    and have a variety of promising RNG's at your disposal.  */
>
> /* You may want use more complicated names for the
>    above simple 1-letter variable names: z,w,x,y,t,c,
>    to avoid clashing with favorites in your code.    */
>
> /* Any one of KISS, MWC, LFIB4, SWB, SHR3, or CONG
>    can be used in an expression to provide a random
>    32-bit integer, and UNI in an expression will
>    provide a random uniform in (01), or VNI in (-1,1).
>    For example, for int i, float v; i=(MWC>>24); will
>    provide a random byte, while v=4.+3.*UNI; will
>    provide a uniform v in the interval 4. to 7.
>    For the super cautious, (KISS+SWB) in an expression
>    would provide a random 32-bit integer from
>    a sequence with period > 2^7700, and would only
>    add some 300 nanoseconds to the computing
>    time for that expression.                        */
>
> /* The KISS generator, (Keep It Simple Stupid), is
>    designed to combine the two multiply-with-carry
>    generators in MWC with the 3-shift register SHR3
>    and the congruential generator CONG, using
>    addition and exclusive-or. Period about 2^123. It
>    is one of my favorite generators.   */
>
> /* The  MWC generator concatenates two 16-bit
>   multiply-with-carry generators, x(n)=36969x(n-1)+carry,
>   y(n)=18000y(n-1)+carry  mod 2^16, has  period  about
>   2^60 and seems to pass all tests of randomness. A favorite
>   stand-alone generator---faster than KISS, which contains it.*/
>
> /* SHR3 is a 3-shift-register generator with
>    period 2^32-1. It uses
>    y(n)=y(n-1)(I+L^17)(I+R^13)(I+L^5), with the
>    y's viewed as binary vectors, L the 32x32
>    binary matrix that shifts a vector left 1, and
>    R its transpose.  SHR3 seems to pass all except
>    the binary rank test, since 32 successive
>    values, as binary vectors, must be linearly
>    independent, while 32 successive truly random
>    32-bit integers, viewed as binary vectors, will
>    be linearly independent only about 29% of the time.   */
>
> /* CONG is a congruential generator with the
>    widely used 69069 as multiplier:
>    x(n)=69069x(n-1)+1234567.  It has period 2^32.
>    The leading half of its 32 bits seem to pass
>    all tests, but bits in the last half are too
>    regular.                              */
>
> /* LFIB4 is an extension of the class that I have
>    previously defined as  lagged Fibonacci
>    generators: x(n)=x(n-r) op x(n-s), with the x's
>    in a finite set over which there is a binary
>    operation op, such as +,- on integers mod 2^32,
>    * on odd such integers, exclusive-or (xor) on
>    binary vectors. Except for those using
>    multiplication, lagged Fibonacci generators
>    fail various tests of randomness, unless the
>    lags are very long.  To see if more than two
>    lags would serve to overcome the problems of 2-
>    lag generators using +,- or xor, I have
>    developed the 4-lag generator LFIB4:
>    x(n)=x(n-256)+x(n-179)+x(n-119)+x(n-55) mod 2^32.
>    Its period is 2^31*(2^256-1), about 2^287, and
>    it seems to pass all tests---in particular,
>    those of the kind for which 2-lag generators
>    using +,-,xor seem to fail.  For even more
>    confidence in its suitability,  LFIB4 can be
>    combined with KISS, with a resulting period of
>    about 2^410: just use (KISS+LFIB4) in any C
>    expression.                              */
>
> /* SWB is a subtract-with-borrow generator that I
>    developed to give a simple method for producing
>    extremely long periods:
>    x(n)=x(n-222)-x(n-237)-borrow mod 2^32.
>    The 'borrow' is 0 unless set to 1 if computing
>    x(n-1) caused overflow in 32-bit integer
>    arithmetic. This generator has a very long
>    period, 2^7098(2^480-1), about 2^7578. It seems
>    to pass all tests of randomness, but,
>    suspicious of a generator so simple and fast
>    (62 nanosecs at 300MHz), I would suggest
>    combining SWB with KISS, MWC, SHR3, or CONG. */
>
> /* Finally, because many simulations call for
```

```
>    uniform random variables in 0&LT;v&LT;1 or -1&LT;v&LT;1, I
>    use #define statements that permit inclusion of
>    such variates directly in expressions:  using
>    UNI will provide a uniform random real (float)
>    in (0,1), while VNI will provide one in (-1,1).  */
>
> /* All of these: MWC, SHR3, CONG, KISS, LFIB4,
>    SWB, UNI and VNI, permit direct insertion of
>    the desired random quantity into an expression,
>    avoiding the time and space costs of a function
>    call. I call these in-line-define functions.
>    To use them, static variables z,w,jsr and
>    jcong should be assigned seed values other than
>    their initial values.  If LFIB4 or SWB are
>    used, the static table t[256] must be
>    initialized.  A sample procedure follows. */
>
> /* A note on timing:  It is difficult to provide
>    exact time costs for inclusion of one of these
>    in-line-define functions in an expression.
>    Times may differ widely for different
>    compilers, as the C operations may be deeply
>    nested and tricky. I suggest these rough
>    comparisons, based on averaging ten runs of a
>    routine that is essentially a long loop:
>    for(i=1;i&LT;10000000;i++) L=KISS; then with KISS
>    replaced with SHR3, CONG,... or KISS+SWB, etc.
>    The times on my home PC, a Pentium 300MHz, in
>    nanoseconds: LFIB4=64; CONG=90; SWB=100;
>    SHR3=110; KISS=209; KISS+LFIB4=252; KISS+SWB=310.     */
```

Subject: Re: Random numbers in C: Some suggestions.
Date: Tue, 12 Jan 1999 13:56:41 -0500
From: George Marsaglia <geo@stat.fsu.edu>
Message-ID: <369B9AE9.52C98810@stat.fsu.edu>
References: <369B66AB.6CDED9F8@ix.netcom.com>
Newsgroups: sci.stat.math,sci.math,sci.math.num-analysis,sci.crypt,sci.physics
Lines: 59

Charles Bond wrote:

```
> Thanks for the post. I want to comment on the SWB routine. I've been
> using
> a similar routine in high speed simulations for years. ...

&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&
     Many years?  It hasn't been very many years
      since I invented the subtract-with-borrow method,
      and developed theory for establishing the periods.
&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&

>   . I believe Knuth briefly discussed the method with guarded
> approval -- constrained by the concern that there was no real theory
> behind it. Do you know if any theoretical work has been done since
> Knuth's book to justify SWB?

> &&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&
>    This business of providing 'theoretical support' for
>   RNG's tends to be overdone---perhaps
>   because of the influence of Knuth. His marvelous
>   expository and poweful mathematical skills have
>   justifiably made him the leading authority.
>   Many people do not understand that such theory
>   is based on the simple fact that congruential
>   random numbers "fall mainly in the planes",
>   that is,  points whose coordinates are succesive
>   elements from the sequence lie on a lattice
>   with a huge unit-cell volume, (m^(n-1) in n-dimensions),
>   compared to the  unit-cell volume of 1 for truly random
>   integers.  So the "lattice test", as I called it,
>   applies to congruential generators, although the ideas have
>   been extended to the near-lattice-like patterns of
>   certain other kinds of generators.  But there seem
>   to be no such lattice-like patterns for many other
>   kinds of generators, and even if there were, it
>   is an easy matter to destroy such  patterns by
>   combining with generators having disparate mathematical
>   structures.
>
>   The quote from my Keynote Address at the 1984
>   Computer Science and Statistics: Symposium on the
>   Interface,  still applies:
>
>   "A random number generator is like sex:
>      When it's good, its wonderful;
>      And when it's bad, it's still pretty good."
>
>   Add to that, in line with my recommendations
```

```
>   on combination generators;
>
>     "And if it's bad, try a twosome or threesome."
>
> George Marsaglia
```

Subject: Re: Random numbers in C: Some suggestions.
Date: Tue, 12 Jan 1999 16:09:29 -0800
From: Charles Bond <cbond@ix.netcom.com>
Message-ID: <369BE439.92E0E011@ix.netcom.com>
References: <369B9AE9.52C98810@stat.fsu.edu>
Newsgroups: sci.stat.math,sci.math,sci.math.num-analysis,sci.crypt,sci.physics
Lines: 69

For the record, I did not mean to imply that Knuth's subtractive generator was *the same* as your subtract with borrow, only that it was *similar* (high speed, no multiplications). I gladly acknowledge your claim on it. But you seem a little skeptical of it yourself, and I was just curious.

Regards,

C. Bond

George Marsaglia wrote:

```
> Charles Bond wrote:
>
> > Thanks for the post. I want to comment on the SWB routine. I've been
> > using
> > a similar routine in high speed simulations for years. ...
>
> &&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&
>      Many years?  It hasn't been very many years
>       since I invented the subtract-with-borrow method,
>       and developed theory for establishing the periods.
> &&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&
>
> >   . I believe Knuth briefly discussed the method with guarded
> > approval -- constrained by the concern that there was no real theory
> > behind it. Do you know if any theoretical work has been done since
> > Knuth's book to justify SWB?
>
> > &&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&
> >    This business of providing 'theoretical support' for
> >   RNG's tends to be overdone---perhaps
> >   because of the influence of Knuth. His marvelous
> >   expository and poweful mathematical skills have
> >   justifiably made him the leading authority.
> >   Many people do not understand that such theory
> >   is based on the simple fact that congruential
> >   random numbers "fall mainly in the planes",
> >   that is,  points whose coordinates are succesive
> >   elements from the sequence lie on a lattice
> >   with a huge unit-cell volume, (m^(n-1) in n-dimensions),
> >   compared to the  unit-cell volume of 1 for truly random
> >   integers.  So the "lattice test", as I called it,
> >   applies to congruential generators, although the ideas have
> >   been extended to the near-lattice-like patterns of
> >   certain other kinds of generators.  But there seem
> >   to be no such lattice-like patterns for many other
> >   kinds of generators, and even if there were, it
> >   is an easy matter to destroy such  patterns by
> >   combining with generators having disparate mathematical
> >   structures.
> >
> >   The quote from my Keynote Address at the 1984
> >   Computer Science and Statistics: Symposium on the
> >   Interface,  still applies:
> >
> >   "A random number generator is like sex:
> >      When it's good, its wonderful;
> >      And when it's bad, it's still pretty good."
> >
> >   Add to that, in line with my recommendations
> >   on combination generators;
> >
> >     "And if it's bad, try a twosome or threesome."
> >
> > George Marsaglia
```

Subject: Re: Random numbers in C: Some suggestions.
Date: Tue, 12 Jan 1999 15:54:10 -0800
From: Mike Oliver <oliver@math.ucla.edu>
Message-ID: <369BE0A2.CA197F7B@math.ucla.edu>
References: <369B5E30.65A55FD1@stat.fsu.edu>

Newsgroups: sci.stat.math,sci.math,sci.math.num-analysis,sci.crypt,sci.physics
Lines: 17

George Marsaglia wrote:

> [...] and experience has shown that
> lagged Fibonacci generators using xor provide
> unsatisfactory 'randomness' unless the lags are
> very long, and even for those with very long lags,
> (and even for those using + or - rather than xor),

Could you give us a pointer to information about
why these RNGs are unsatisfactory and what sort
of test they tend to fail?

--
Disclaimer: I could be wrong -- but I'm not. (Eagles, "Victim of Love")

Finger for PGP public key, or visit http://www.math.ucla.edu/~oliver.
1500 bits, fingerprint AE AE 4F F8 EA EA A6 FB E9 36 5F 9E EA D0 F8 B9

Subject: Re: Random numbers in C: Some suggestions.
Date: 13 Jan 1999 12:21:37 -0800
From: Eric Backus <ericb@labejb.lsid.hp.com>
Message-ID: <tun23mnctq.fsf@labejb.lsid.hp.com>
References: <369B5E30.65A55FD1@stat.fsu.edu>
Newsgroups: sci.stat.math,sci.math,sci.math.num-analysis,sci.crypt,sci.physics
Lines: 65

George Marsaglia <geo@stat.fsu.edu> writes:

> This posting ends with 17 lines of
> C code that provide eight different
> in-line random number generators, six for
> random 32-bit integers and two for uniform
> reals in (0,1) and (-1,1).
> Comments are interspersed with that
> code. Various combinations of the six in-line
> integer generators may put in C expressions to
> provide a wide variety of very fast, long period,
> well-tested RNG's. I invite comments, feedback,
> verifications and timings.

```
> #define UL unsigned long
> #define znew  ((z=36969*(z&65535)+(z>>16))&LT;&LT;16)
> #define wnew  ((w=18000*(w&65535)+(w>>16))&65535)
> #define MWC   (znew+wnew)
> #define SHR3  (jsr=(jsr=(jsr=jsr^(jsr&LT;&LT;17))^(jsr>>13))^(jsr&LT;&LT;5))
> #define CONG  (jcong=69069*jcong+1234567)
> #define KISS  ((MWC^CONG)+SHR3)
> #define LFIB4 (t[c]=t[c]+t[c+58]+t[c+119]+t[++c+178])
> #define SWB   (t[c+237]=(x=t[c+15])-(y=t[++c]+(x&LT;y)))
> #define UNI   (KISS*2.328306e-10)
> #define VNI   ((long) KISS)*4.656613e-10
> /*  Global static variables: */
>  static UL z=362436069, w=521288629, jsr=123456789, jcong=380116160;
>  static UL t[256];
>  static UL x=0,y=0; static unsigned char c=0;
>
> /* Random seeds must be used to reset z,w,jsr,jcong and
> the table t[256]  Here is an example procedure, using KISS: */
>
>  void settable(UL i1,UL i2,UL i3,UL i4)
>  { int i; z=i1;w=i2,jsr=i3; jcong=i4;
>  for(i=0;i&LT;256;i++)  t[i]=KISS;        }
```

Thank you for providing this extremely useful code. (I'd like to make
use of it, however I see no copyright notice, can I assume you are
making it free for anyone to use?)

I have a small problem with the definition of LFIB4 and SWB. In an
attempt to make these a single line of C code, they both use "++c" in
the same expression as they use "c". A C compiler is free to
rearrange the order in which it calculates the intermediate terms of
these expressions, so the expressions can produce different results
depending on the compiler.

I might propose alternate expressions using the "," operator in
order to remove any ambiguity. With a good compiler, these
expressions probably won't be any slower than your original ones:

```
#define LFIB4_ALT (t[c]=t[c]+t[c+58]+t[c+119]+t[c+179],t[c++])
#define SWB_ALT   (t[c+237]=(x=t[c+15])-(y=t[c+1]+(x&LT;y)),t[c++ +237])
```

However, these are uglier and harder to understand than your original
expressions, and of course I might have made a mistake in interpreting
where the c++ should go. Any comments?

--
Eric Backus <eric_backus@hp.com>
http://labejb.lsid.hp.com/

(425) 335-2495

Subject: Random numbers for C: Improvements.
Date: Fri, 15 Jan 1999 11:41:47 -0500
From: George Marsaglia <geo@stat.fsu.edu>
Message-ID: <369F6FCA.74C7C041@stat.fsu.edu>
References: <369B5E30.65A55FD1@stat.fsu.edu>
Newsgroups: sci.stat.math,sci.math,sci.math.numer-analysis
Lines: 93

As I hoped, several suggestions have led to improvements in the code for RNG's I proposed for use in C. (See the thread "Random numbers for C: Some suggestions" in previous postings.) The improved code is listed below.

A question of copyright has also been raised. Unlike DIEHARD, there is no copyright on the code below. You are free to use it in any way you want, but you may wish to acknowledge the source, as a courtesy.

To avoid being cited by the style police, some have suggested using typedef rather than #define in order to replace unsigned long by UL.

Others have pointed out that one cannot be certain of the way that a compiler will evaluate terms in a sum, so using ++c in a term is dangerous. They have offered a version using the comma operator, which ensures that the table index is incremented properly. See LFIB4 and SWB below.

In my earlier work, done in Fortran, I had implemented two 16-bit multiply-with-carry generators, say z and w, as 32-bit integers, with the carry in the top 16 bits, the output in the bottom 16. They were combined by (z<<16)+w. (In Fortran, ishft(z,16)+w.) Such a combination seemed to pass all tests. In the above-mentioned post, I used (z<<16)+(w&65525), and that does not pass all tests. So (z<<16)+w seems preferable; it is used below, providing a MWC that seems to pass all tests.

The generators MWC, KISS, LFIB4 and SWB seem to pass all tests. By themselves, CONG and SHR3 do not, but using CONG+SHR3 provides one of the fastest combinations that satisfy the DIEHARD battery of tests.

Of course, one cannot have absolute confidence in any generator. The choices LFIB4 and SWB have immense periods, are very fast, and pass all tests in DIEHARD, but I am hesitant to rely on them alone---primarily because they come from such simple mod 2^32 arithmetic: four adds in LFIB4 or one subtract-with-borrow in SWB.

The code below provides a variety of in-line generators that seem promising by themselves, and even more so in combination. With them, one may feel fairly confident that combinations will produce results consistent with the underlying probability theory in your applications.

All combinations seem to support the supplemented quote from my 1984 Keynote Address:

> A random number generator is like sex;
> When it's good, it's wonderful,
> And when it's bad, it's still pretty good.
>
> And when it's bad, try a twosome or threesome.

The C code follows; you may want to snip and save from here----------:

```
#define znew  (z=36969*(z&65535)+(z>>16))
#define wnew  (w=18000*(w&65535)+(w>>16))
#define MWC   ( (znew&LT;&LT;16)+wnew )
#define SHR3  (jsr=(jsr=(jsr=jsr^(jsr&LT;&LT;17))^(jsr>>13))^(jsr&LT;&LT;5))
#define CONG  (jcong=69069*jcong+1234567)
#define KISS  ((MWC^CONG)+SHR3)
#define LFIB4 (t[c]=t[c]+t[c+58]+t[c+119]+t[c+178],t[++c])
#define SWB   (t[c+237]=(x=t[c+15])-(y=t[c]+(x&LT;y)),t[++c])
#define UNI   (KISS*2.328306e-10)
#define VNI   ((long) KISS)*4.656613e-10
typedef unsigned long UL;

/*  Global static variables: */
 static UL z=362436069, w=521288629, jsr=123456789, jcong=380116160;
 static UL t[256];
 static UL x=0,y=0; static unsigned char c=0;
/* Random seeds must be used to reset z,w,jsr,jcong and
the table t[256].  Here is an example procedure, using KISS: */
```

```
 void settable(UL i1,UL i2,UL i3,UL i4)
 { int i; z=i1;w=i2,jsr=i3; jcong=i4;
 for(i=0;i&LT;256;i++)  t[i]=KISS;        }

/* Any one of KISS, MWC, LFIB4, SWB, SHR3, or CONG  can be used in
   an expression to provide a random 32-bit integer, while UNI
   provides a real in (0,1) and VNI a real in (-1,1).   */
```

Subject: Re: Random numbers for C: Improvements.
Date: 15 Jan 1999 14:02:47 -0500
From: zaykin@statgen.ncsu.edu (Dmitri Zaykin)
Message-ID: <790f49x60.fsf@poole.statgen.ncsu.edu>
References: <369F6FCA.74C7C041@stat.fsu.edu>
Newsgroups: sci.stat.math,sci.math,sci.math.num-analysis
Lines: 11

George Marsaglia <geo@stat.fsu.edu> writes:

```
> #define LFIB4 (t[c]=t[c]+t[c+58]+t[c+119]+t[c+178],t[++c])
> #define SWB   (t[c+237]=(x=t[c+15])-(y=t[c]+(x&LT;y)),t[++c])
```

Shouldn't these be

```
#define LFIB4 (t[c]=t[c]+t[c+58]+t[c+119]+t[c+179], ++c)
#define SWB   (t[c+237]=(x=t[c+15])-(y=t[c+1]+(x&LT;y)), ++c)
```

Dmitri

Subject: Re: Random numbers for C: Improvements.
Date: 15 Jan 99 20:16:13 GMT
From: radford@cs.toronto.edu (Radford Neal)
Message-ID: <99Jan15.151547edt.785@neuron.ai.toronto.edu>
References: <790f49x60.fsf@poole.statgen.ncsu.edu>
Newsgroups: sci.math,sci.math.num-analysis,sci.stat.math
Lines: 28

>George Marsaglia <geo@stat.fsu.edu> writes:
>
>> #define LFIB4 (t[c]=t[c]+t[c+58]+t[c+119]+t[c+178],t[++c])
>> #define SWB   (t[c+237]=(x=t[c+15])-(y=t[c]+(x<y)),t[++c])
>

In article <790f49x60.fsf@poole.statgen.ncsu.edu>,
Dmitri Zaykin <zaykin@statgen.ncsu.edu> wrote:

>Shouldn't these be
>
>#define LFIB4 (t[c]=t[c]+t[c+58]+t[c+119]+t[c+179], ++c)
>#define SWB   (t[c+237]=(x=t[c+15])-(y=t[c+1]+(x<y)), ++c)
>

This doesn't work either. I believe that it is undefined whether the comparison x<y uses the new or the old value of x. The SHR3 macro in the original source also suffers from this flaw.

I think one needs to face up to an unpleasant truth: The #define facility of C was poorly designed, and is incapable in general of supporting the definition of "in-line" procedures. It is far better to simply write ordinary C procedures, and accept the fairly small procedure call overhead.

   Radford Neal

Subject: Re: Random numbers for C: Improvements.
Date: Fri, 15 Jan 1999 17:34:36 -0600
From: "Jeff Stout" <jstout@ncon.com>
Message-ID: <QfQn2.10695$bf6.2024@news1.giganews.com>
References: <99Jan15.151547edt.785@neuron.ai.toronto.edu>
Newsgroups: sci.math,sci.math.num-analysis,sci.stat.math
Lines: 27

Radford Neal wrote in message
<99Jan15.151547edt.785@neuron.ai.toronto.edu>...
>
>
>I think one needs to face up to an unpleasant truth: The #define
>facility of C was poorly designed, and is incapable in general of
>supporting the definition of "in-line" procedures. It is far better
>to simply write ordinary C procedures, and accept the fairly small
>procedure call overhead.
>
>   Radford Neal
>
>

This is not a poor design of the macro facility of C, but a built in
limitation of the C language itself. The preprocessor is just doing
a stupid text substituation, its the C compiler that is ambigous about
the interpretation

The C macro language can support "in-line" procedures and a lot
of other junk only true software wienies would ever use.

Jeff Stout

Subject: Re: Random numbers for C: Improvements.
Date: 16 Jan 99 04:43:12 GMT
From: radford@cs.toronto.edu (Radford Neal)
Message-ID: <99Jan15.233424edt.6005@cortex.ai.toronto.edu>
References: <QfQn2.10695$bf6.2024@news1.giganews.com>
Newsgroups: sci.math,sci.math.num-analysis,sci.stat.math
Lines: 88

>Radford Neal wrote in message
><99Jan15.151547edt.785@neuron.ai.toronto.edu>...
>>
>>I think one needs to face up to an unpleasant truth: The #define
>>facility of C was poorly designed, and is incapable in general of
>>supporting the definition of "in-line" procedures. It is far better
>>to simply write ordinary C procedures, and accept the fairly small
>>procedure call overhead.

In article <QfQn2.10695$bf6.2024@news1.giganews.com>,
Jeff Stout <jstout@ncon.com> wrote:
>This is not a poor design of the macro facility of C, but a built in
>limitation of the C language itself. The preprocessor is just doing
>a stupid text substituation, its the C compiler that is ambigous about
>the interpretation
>
>The C macro language can support "in-line" procedures and a lot
>of other junk only true software wienies would ever use.

In saying that the #define facility of C was poorly designed, I of
course meant that it does not do what is needed to write good programs.
The fact that it does stupid text substitation is a big part of this.

In the case under discussion, the only reason (I hope!) that obscure
and semantically undefined constructs were being used was in order to
try to define an in-line procedure using this inadequate facility.

Unfortunately, although one can indeed create lots of junk using the
C pre-processor, one cannot, in general, use it to create in-line
procedures. One can't write a macro equivalent to the following,
for instance:

```
  int first_zero (int *a)
  { int i;
    for (i = 0; a[i]!=0; i++) ;
    return i;
  }
```

Furthermore, in many cases where one CAN write in-line procedures
using C macros, 99% of the people who try to do so get it wrong. For
example, the following procedure CAN be written as a macro:

```
  void badexample(int a)
  {
    while (f(a)==0) ;
    g(a);
  }
```

but NOT as follows:

```
  #define badexample(a) while (f(a)==0) ; g(a);
```

nor as follows:

```
  #define badexample(a) { while (f(a)==0) ; g(a); }
```

The following gets closer:

```
  #define badexample(a) do { while (f(a)==0) ; g(a); } while (0)
```

Note: NO semicolon at the end. The necessity of this trick is left
as an exercise for the reader.

Really, though, one needs to define the macro as follows:

```
  #define badexample(a) do { int fy7653_4xq = a; \
                              while (f(fy7653_4xq)==0) ; \
                              g(fy7653_4xq); \
                          } while (0)
```

Here, fy7853_4xq is assumed to be sufficiently obscure that it is
unlikely to conflict with the name of a variable that occurs in a.

Not illustrated in this example, but commonly overlooked, is the frequent necessity to enclose references to the parameters of the macro in parentheses.

Failure to understand these arcane necessities will result in macros that behave just like the procedure they are trying to mimic around 95% of the times they are called, and which produce completely obscure bugs the other 5% of the time.

Radford Neal

Subject: Re: Random numbers for C: Improvements.
Date: 16 Jan 1999 16:14:45 -0500
From: zaykin@statgen.ncsu.edu (Dmitri Zaykin)
Message-ID: <7ogny6htm.fsf@poole.statgen.ncsu.edu>
References: <QfQn2.10695$bf6.2024@news1.giganews.com>
Newsgroups: sci.math,sci.math.num-analysis,sci.stat.math
Lines: 15

"Jeff Stout" <jstout@ncon.com> writes:

> The C macro language can support "in-line" procedures and a lot
> of other junk only true software wienies would ever use.

It is probably true that the posted random number generators are better implemented as regular functions. C compilers should attempt to inline "short" functions into their callers when certain optimization flags are enabled.

Also, the new C standard will support the "inline" keyword. Some compilers (e.g. gcc called without "-ansi" switch) already recognize it.

Dmitri

Subject: Re: Random numbers for C: Improvements.
Date: 16 Jan 1999 15:19:24 -0500
From: zaykin@statgen.ncsu.edu (Dmitri Zaykin)
Message-ID: <7pv8f55tf.fsf@poole.statgen.ncsu.edu>
References: <99Jan15.151547edt.785@neuron.ai.toronto.edu>
Newsgroups: sci.math,sci.math.num-analysis,sci.stat.math
Lines: 29

These should work (aside the possibility of clashing with local names)

```
#define SHR3  (jsr=jsr^(jsr&LT;&LT;17), jsr=jsr^(jsr>>13), jsr=jsr^(jsr&LT;&LT;5))
#define LFIB4 (t[c] = t[c]+t[c+58]+t[c+119]+t[c+179], t[++c])
#define SWB   (x = t[c+15], t[c+237] = x-(y=t[c+1]+(x&LT;y)), t[++c])
```

Dmitri

radford@cs.toronto.edu (Radford Neal) writes:

```
> >George Marsaglia &LT;geo@stat.fsu.edu> writes:
> >
> >> #define LFIB4 (t[c]=t[c]+t[c+58]+t[c+119]+t[c+178],t[++c])
> >> #define SWB   (t[c+237]=(x=t[c+15])-(y=t[c]+(x&LT;y)),t[++c])
> >
>
> In article &LT;790f49x60.fsf@poole.statgen.ncsu.edu>,
> Dmitri Zaykin &LT;zaykin@statgen.ncsu.edu> wrote:
>
> >Shouldn't these be
> >
> >#define LFIB4 (t[c]=t[c]+t[c+58]+t[c+119]+t[c+179], ++c)
> >#define SWB   (t[c+237]=(x=t[c+15])-(y=t[c+1]+(x&LT;y)), ++c)
> >
>
> This doesn't work either.  I believe that it is undefined whether the
> comparison x&LT;y uses the new or the old value of x.  The SHR3 macro
> in the original source also suffers from this flaw.
```

Subject: Re: Random numbers for C: Improvements.
Date: 17 Jan 1999 08:06:29 PST
From: Ramin Sina <rsina@concentric.net>
Message-ID: <36A21A2C.AD5145D1@concentric.net>
References: <7pv8f55tf.fsf@poole.statgen.ncsu.edu>
Newsgroups: sci.math,sci.math.num-analysis,sci.stat.math
Lines: 34

Dmitri Zaykin wrote:

> These should work (aside the possibility of clashing with local names)

```
>
> #define SHR3  (jsr=jsr^(jsr&LT;&LT;17), jsr=jsr^(jsr>>13), jsr=jsr^(jsr&LT;&LT;5))
> #define LFIB4 (t[c] = t[c]+t[c+58]+t[c+119]+t[c+179], t[++c])
> #define SWB   (x = t[c+15], t[c+237] = x-(y=t[c+1]+(x&LT;y)), t[++c])
>
> Dmitri
>
> radford@cs.toronto.edu (Radford Neal) writes:
>
> > >George Marsaglia &LT;geo@stat.fsu.edu> writes:
> > >
> > >> #define LFIB4 (t[c]=t[c]+t[c+58]+t[c+119]+t[c+178],t[++c])
> > >> #define SWB   (t[c+237]=(x=t[c+15])-(y=t[c]+(x&LT;y)),t[++c])
> > >
> >
> >
```

Could someone please post a short example code on how to use these in practice. I am lost in the discussion in this thread, but I would like very much to understand how to use these for example to generate 1000 uniform random numbers between betwen A and B (say 5 and 10) .

Thanks very much,
Ramin Sina

```
--
-----------------------------------------------------------
 Ramin Sina                          rsina@concentric.net
```

```
Subject: Re: Random numbers for C: Improvements.
Date: 18 Jan 1999 17:34:52 GMT
From: hook@nas.nasa.gov (Ed Hook)
Message-ID: &LT;77vrbs$pmj$1@sun500.nas.nasa.gov>
References: &LT;7pv8f55tf.fsf@poole.statgen.ncsu.edu>
Newsgroups: sci.math,sci.math.num-analysis,sci.stat.math
Lines: 25

In article &LT;7pv8f55tf.fsf@poole.statgen.ncsu.edu>,
               zaykin@statgen.ncsu.edu (Dmitri Zaykin) writes:
|> These should work (aside the possibility of clashing with local names)
|>
|> #define SHR3  (jsr=jsr^(jsr&LT;&LT;17), jsr=jsr^(jsr>>13), jsr=jsr^(jsr&LT;&LT;5))
|> #define LFIB4 (t[c] = t[c]+t[c+58]+t[c+119]+t[c+179], t[++c])
|> #define SWB   (x = t[c+15], t[c+237] = x-(y=t[c+1]+(x&LT;y)), t[++c])
|>
|> Dmitri
|>

   No -- 'SHR3' and 'LFIB4' are OK, but your version of 'SWB'
   still invokes the dreaded undefined behavior. In the second
   calculation, there's no sequence point to separate the use
   of 'y' to determine the value of 'x&LT;y' and the assignment to 'y'.
   One might argue that this is not particularly relevant, since
   we're trying to simulate randomness anyhow ... except that, once
   undefined behavior rears its ugly head, the code is allowed to
   simply dump core and die ...

--
 Ed Hook                              |       Copula eam, se non posit
 MRJ Technology Solutions, Inc.       |         acceptera jocularum.
 NAS, NASA Ames Research Center       | I can barely speak for myself, much
 Internet: hook@nas.nasa.gov          |         less for my employer
```

```
Subject: Re: Random numbers for C: Improvements.
Date: 19 Jan 1999 13:10:13 GMT
From: orjanjo@math.ntnu.no (Orjan Johansen)
Message-ID: &LT;78207l$1l8$1@due.unit.no>
References: &LT;77vrbs$pmj$1@sun500.nas.nasa.gov>
Newsgroups: sci.math,sci.math.num-analysis,sci.stat.math
Lines: 21

In article &LT;77vrbs$pmj$1@sun500.nas.nasa.gov>,
Ed Hook &LT;hook@nas.nasa.gov> wrote:
>In article &LT;7pv8f55tf.fsf@poole.statgen.ncsu.edu>,
>               zaykin@statgen.ncsu.edu (Dmitri Zaykin) writes:
[snip]
>|> #define SWB   (x = t[c+15], t[c+237] = x-(y=t[c+1]+(x&LT;y)), t[++c])
[snip]
>   No -- 'SHR3' and 'LFIB4' are OK, but your version of 'SWB'
>   still invokes the dreaded undefined behavior. In the second
>   calculation, there's no sequence point to separate the use
>   of 'y' to determine the value of 'x&LT;y' and the assignment to 'y'.
[snip]

I am not a programmer by trade, but I have a little knowledge of C.
Surely in an expression of the form y=..., the evaluation of ... is
guaranteed to happen before the assignment.  As far as I understand,
what is not guaranteed is that any assignments in ... would happen
before the one to y.
```

Greetings,
Ørjan.

Subject: Re: Random numbers for C: Improvements.
Date: Tue, 19 Jan 1999 18:29:34 GMT
From: horst.kraemer@snafu.de (Horst Kraemer)
Message-ID: <36a4cb5e.39101741@news.snafu.de>
References: <77vrbspmj1@sun500.nas.nasa.gov>
Newsgroups: sci.math,sci.math.num-analysis,sci.stat.math
Lines: 39

On 18 Jan 1999 17:34:52 GMT, hook@nas.nasa.gov (Ed Hook) wrote:

> In article <7pv8f55tf.fsf@poole.statgen.ncsu.edu>,
> zaykin@statgen.ncsu.edu (Dmitri Zaykin) writes:
> |> These should work (aside the possibility of clashing with local names)
> |>
> |> #define SHR3 (jsr=jsr^(jsr<<17), jsr=jsr^(jsr>>13),
jsr=jsr^(jsr<<5))
> |> #define LFIB4 (t[c] = t[c]+t[c+58]+t[c+119]+t[c+179], t[++c])
> |> #define SWB   (x = t[c+15], t[c+237] = x-(y=t[c+1]+(x<y)), t[++c])
> |>
> |> Dmitri
> |>
>
> No -- 'SHR3' and 'LFIB4' are OK, but your version of 'SWB'
> still invokes the dreaded undefined behavior. In the second
> calculation, there's no sequence point to separate the use
> of 'y' to determine the value of 'x<y' and the assignment to 'y'.

Sorry. No sequence point is needed between reading an writing to an
lvalue.

According to your theory

```
        y = y - 1
```

would produce undefined behaviour, too ....

What produces undefined behaviour in C is _modifying_ an lvalue more
than once between the last and next sequence point. There is only one
modification of y in the expression

```
         t[c+237] = x-(y=t[c+1]+(x&LT;y))
```

Regards
Horst

Subject: Re: Random numbers for C: Improvements.
Date: 17 Jan 1999 13:22:21 -0500
From: hrubin@b.stat.purdue.edu (Herman Rubin)
Message-ID: <77t9ot$15ha@b.stat.purdue.edu>
References: <99Jan15.151547edt.785@neuron.ai.toronto.edu>
Newsgroups: sci.math,sci.math.num-analysis,sci.stat.math
Lines: 49

In article <99Jan15.151547edt.785@neuron.ai.toronto.edu>,
Radford Neal <radford@cs.toronto.edu> wrote:
>>George Marsaglia <geo@stat.fsu.edu> writes:

>>> #define LFIB4 (t[c]=t[c]+t[c+58]+t[c+119]+t[c+178],t[++c])
>>> #define SWB   (t[c+237]=(x=t[c+15])-(y=t[c]+(x<y)),t[++c])

>In article <790f49x60.fsf@poole.statgen.ncsu.edu>,
>Dmitri Zaykin <zaykin@statgen.ncsu.edu> wrote:

>>Shouldn't these be

>>#define LFIB4 (t[c]=t[c]+t[c+58]+t[c+119]+t[c+179], ++c)
>>#define SWB   (t[c+237]=(x=t[c+15])-(y=t[c+1]+(x<y)), ++c)

>This doesn't work either. I believe that it is undefined whether the
>comparison x<y uses the new or the old value of x. The SHR3 macro
>in the original source also suffers from this flaw.

>I think one needs to face up to an unpleasant truth: The #define
>facility of C was poorly designed, and is incapable in general of
>supporting the definition of "in-line" procedures. It is far better
>to simply write ordinary C procedures, and accept the fairly small
>procedure call overhead.

I think it should be clarified, and probably written out in some
more detail. But the procedure call overhead would be comparable
to the computing cost; C, and all other languages, have such great
built-in inefficiencies that what is needed is something written

from the standpoint of mathematics and efficiency.

But even the use of a comparison is a costly operation, if the
result is to be used promptly. Any form of random, pseudo-random,
or quasi-random numbers should be done in good integer form,
not present now on many machines because of bad design coming
from bad languages which have thrown out the natural properties
of computers, and attempts to keep stupid people from making
mistakes. An idiot-proof language is only for idiots.

--
This address is for information only. I do not claim that these views
are those of the Statistics Department or of Purdue University.
Herman Rubin, Dept. of Statistics, Purdue Univ., West Lafayette IN47907-1399
hrubin@stat.purdue.edu Phone: (765)494-6054 FAX: (765)494-0558

Subject: Re: Random numbers for C: Improvements.
Date: Sun, 17 Jan 1999 19:10:01 GMT
From: dmurdoch@pair.com (Duncan Murdoch)
Message-ID: <36a23371.10418803@newshost.uwo.ca>
References: <77t9ot$15ha@b.stat.purdue.edu>
Newsgroups: sci.stat.math
Lines: 27

On 17 Jan 1999 13:22:21 -0500, hrubin@b.stat.purdue.edu (Herman Rubin)
wrote:
> But the procedure call overhead would be comparable
>to the computing cost;

I don't know if that's true or not, but I don't think it is really
something to worry about. The uniform value coming out of these
macros is unlikely to be the end of the computation; you'll almost
certainly do far more costly things with it after it's generated. So
halving or doubling the speed of the generator won't have nearly so
much impact on the overall calculation.

On the other hand, if it's programmed in a way that is sometimes
incorrect in hard to detect ways, it might invalidate the whole
calculation. That seems like a much higher cost to pay.

>An idiot-proof language is only for idiots.

No, if such a thing existed, it would also reduce the occasional
lapses of non-idiots. Programmers are fallible; if tools can prevent
some common errors (even if it means doubling the computation time),
then those tools are worth using.

I'd rather not be the first one to finish, if my answer is wrong!

Duncan Murdoch

Subject: Re: Random numbers for C: Improvements.
Date: 18 Jan 1999 08:56:33 -0500
From: hrubin@b.stat.purdue.edu (Herman Rubin)
Message-ID: <77veih$11ts@b.stat.purdue.edu>
References: <36a23371.10418803@newshost.uwo.ca>
Newsgroups: sci.stat.math
Lines: 63

In article <36a23371.10418803@newshost.uwo.ca>,
Duncan Murdoch <dmurdoch@pair.com> wrote:
>On 17 Jan 1999 13:22:21 -0500, hrubin@b.stat.purdue.edu (Herman Rubin)
>wrote:
>> But the procedure call overhead would be comparable
>>to the computing cost;

>I don't know if that's true or not, but I don't think it is really
>something to worry about. The uniform value coming out of these
>macros is unlikely to be the end of the computation; you'll almost
>certainly do far more costly things with it after it's generated. So
>halving or doubling the speed of the generator won't have nearly so
>much impact on the overall calculation.

It certainly is. The cost of one procedure call could far exceed
the cost of a uniform, or even many nonuniform, random variables.
If procedure calls have to be made in the generation, I would not
be surprised to have a factor of 10 or more. BTW, I would not
use any pseudo-random procedure in any case; XORing the numbers
with physical random numbers, which need not be as good as one
wants the final result to be, should be done, even if those
numbers should be reused, and this should be carefully done.

The fundamental output of a random number generator should be a
bit stream; this can be converted easily to anything, but the
converse is false.

>On the other hand, if it's programmed in a way that is sometimes
>incorrect in hard to detect ways, it might invalidate the whole

>calculation. That seems like a much higher cost to pay.

This is why it is important that those who understand the mathematics of the algorithm do it; I do not believe it pays to have a library routine written by one person. HLL algorithms fail in rather odd ways, and algorithms for generating random numbers, unless they have huge errors, can only be checked by direct proof of correctness.

>>An idiot-proof language is only for idiots.

>No, if such a thing existed, it would also reduce the occasional
>lapses of non-idiots. Programmers are fallible; if tools can prevent
>some common errors (even if it means doubling the computation time),
>then those tools are worth using.

On the contrary, these tools can cause the programmer to make errors which cannot be easily detected. And it does not mean just a factor of two, but far more. The current design of computers, driven by the languages, has increased the time to do multiple precision arithmetic by a factor of possibly 100 relative to what it would have been if "old" architecture was in place.

>I'd rather not be the first one to finish, if my answer is wrong!

>Duncan Murdoch

--
This address is for information only. I do not claim that these views
are those of the Statistics Department or of Purdue University.
Herman Rubin, Dept. of Statistics, Purdue Univ., West Lafayette IN47907-1399
hrubin@stat.purdue.edu Phone: (765)494-6054 FAX: (765)494-0558

Subject: Re: Random numbers for C: Improvements.
Date: 18 Jan 99 18:38:33 GMT
From: radford@cs.toronto.edu (Radford Neal)
Message-ID: <99Jan18.133644edt.6004@cortex.ai.toronto.edu>
References: <77veih$11ts@b.stat.purdue.edu>
Newsgroups: sci.stat.math
Lines: 60

In article <77veih$11ts@b.stat.purdue.edu>,
Herman Rubin <hrubin@b.stat.purdue.edu> wrote:

>... The cost of one procedure call could far exceed
>the cost of a uniform, or even many nonuniform, random variables.
>If procedure calls have to be made in the generation, I would not
>be surprised to have a factor of 10 or more.

This might have been the case with some old languages such as Algol 60, but not with C as implemented on modern machines. Even on a CISC machine like a VAX, I doubt the C procedure call overhead would be more that twice the time for the actual work, and on modern RISC machines the ratio will tend to be less.

> BTW, I would not
>use any pseudo-random procedure in any case; XORing the numbers
>with physical random numbers, which need not be as good as one
>wants the final result to be, should be done, even if those
>numbers should be reused, and this should be carefully done.

I'd agree with this. I just makes the work inside the procedure greater, however, reducing the significance of the procedure call overhead.

>>>An idiot-proof language is only for idiots.
>
>>No, if such a thing existed, it would also reduce the occasional
>>lapses of non-idiots. Programmers are fallible; if tools can prevent
>>some common errors (even if it means doubling the computation time),
>>then those tools are worth using.
>
>On the contrary, these tools can cause the programmer to make
>errors which cannot be easily detected.

Although such errors caused by use of high-level languages are possible, they are minor compared to the numerous ways one can go wrong when writing assembly-language programs (eg, failing to be completely consistent in conventions for whether procedures do or do not save register contents).

>The current design of
>computers, driven by the languages, has increased the time to
>do multiple precision arithmetic by a factor of possibly 100
>relative to what it would have been if "old" architecture was
>in place.

I find this hard to believe. It's true that high level languages provide no way to access operations such as 32-bit by 32-bit to 64-bit multiplication, but that would just force you to implement multiple precision arithmetic with 16-bit chunks rather than 32-bit chunks, for

a slowdown of "only" a factor of four. Even accounting for additional general inefficiencies, it's hard to see how it can be any more than a factor of 10 slower. If you're claiming that the hardware itself would be designed differently if people still wrote in assembler, I'd say that if anything the effect goes the other way. The tendency with RISC machines has been to simplify the hardware (making it faster) at the expense of funny machine languages, which compilers can cope with, but which are less convenient for humans writing assembly code.

Radford Neal

Subject: Re: Random numbers for C: Improvements.
Date: 19 Jan 1999 10:25:10 -0500
From: hrubin@b.stat.purdue.edu (Herman Rubin)
Message-ID: <78284m$m0e@b.stat.purdue.edu>
References: <99Jan18.133644edt.6004@cortex.ai.toronto.edu>
Newsgroups: sci.stat.math
Lines: 119

In article <99Jan18.133644edt.6004@cortex.ai.toronto.edu>,
Radford Neal <radford@cs.toronto.edu> wrote:
>In article <77veih$11ts@b.stat.purdue.edu>,
>Herman Rubin <hrubin@b.stat.purdue.edu> wrote:

>>... The cost of one procedure call could far exceed
>>the cost of a uniform, or even many nonuniform, random variables.
>>If procedure calls have to be made in the generation, I would not
>>be surprised to have a factor of 10 or more.

>This might have been the case with some old languages such as Algol
>60, but not with C as implemented on modern machines. Even on a CISC
>machine like a VAX, I doubt the C procedure call overhead would be
>more that twice the time for the actual work, and on modern RISC
>machines the ratio will tend to be less.

There are more problems than that; the question is, whether
procedure calls have to be made in the internal computation
of the pseudo-random numbers. In this case, they would require
a huge number of register save-restores. The example considered
using a subroutine call for the generation of each random variable.

The code discussed was not in making a call for an array, but
for a single step in the generation procedure.

>> BTW, I would not
>>use any pseudo-random procedure in any case; XORing the numbers
>>with physical random numbers, which need not be as good as one
>>wants the final result to be, should be done, even if those
>>numbers should be reused, and this should be carefully done.

>I'd agree with this. I just makes the work inside the procedure
>greater, however, reducing the significance of the procedure call
>overhead.

This is best reduced by using buffers.

>>>>An idiot-proof language is only for idiots.

>>>No, if such a thing existed, it would also reduce the occasional
>>>lapses of non-idiots. Programmers are fallible; if tools can prevent
>>>some common errors (even if it means doubling the computation time),
>>>then those tools are worth using.

>>On the contrary, these tools can cause the programmer to make
>>errors which cannot be easily detected.

>Although such errors caused by use of high-level languages are
>possible, they are minor compared to the numerous ways one can go
>wrong when writing assembly-language programs (eg, failing to be
>completely consistent in conventions for whether procedures do or do
>not save register contents).

It depends on how much is done; saving dozens of registers is
certainly expensive. This is the order of magnitude, as the
number of registers on many machines runs from 32 to 256, and
possibly more. What should be used very often instead of calls,
where transfer of control is needed, are jumps with return, or
even quite liberal use of gotos.

>>The current design of
>>computers, driven by the languages, has increased the time to
>>do multiple precision arithmetic by a factor of possibly 100
>>relative to what it would have been if "old" architecture was
>>in place.

>I find this hard to believe. It's true that high level languages
>provide no way to access operations such as 32-bit by 32-bit to 64-bit
>multiplication, but that would just force you to implement multiple
>precision arithmetic with 16-bit chunks rather than 32-bit chunks, for
>a slowdown of "only" a factor of four.

It is 16-bit chunks instead of the size chunks the floating processors
can handle, and also the much smaller parallelism and far slower

multipliers, as well as the problem of handling the much larger number of registers. What will fit in registers for floating is likely to involve a large number of memory references in fixed. There is also likely to be lots of shifting, to extract the smaller chunks.

I did look into it in detail on the Cray, and I needed to use 20 instructions to get a product of two 48-bit numbers.

Even accounting for additional
>general inefficiencies, it's hard to see how it can be any more than a
>factor of 10 slower. If you're claiming that the hardware itself
>would be designed differently if people still wrote in assembler, I'd
>say that if anything the effect goes the other way.

At this stage, this would not be the case. It would probably have been best if, at the time that it was decided to go beyond Fortran, the versatility of the machine instructions at the time had been put into the HLL's. Among those omitted was that of replacements producing lists (NOT structs) of results, and the ability to directly use the same hardware entities directly with different "type" operations. The simplest procedures, from the standpoint of computational complexity, to generate non-uniform random variables from uniform output use bit operations, not floating arithmetic.

The tendency with
>RISC machines has been to simplify the hardware (making it faster) at
>the expense of funny machine languages, which compilers can cope with,
>but which are less convenient for humans writing assembly code.

If we had real RISC, it would not have floating point, as floating can be fairly efficiently done with fixed point. Instead, we would have fixed-point operations yielding multiple results, and flexible hardware. Forced normalization floating point arithmetic is quite difficult to use for multiple precision, or much of anything else. The IEEE committee was notified of these problems.

> Radford Neal

--
This address is for information only. I do not claim that these views are those of the Statistics Department or of Purdue University.
Herman Rubin, Dept. of Statistics, Purdue Univ., West Lafayette IN47907-1399
hrubin@stat.purdue.edu Phone: (765)494-6054 FAX: (765)494-0558

Subject: Re: Random numbers for C: Improvements.
Date: Tue, 19 Jan 1999 03:23:26 GMT
From: dmurdoch@pair.com (Duncan Murdoch)
Message-ID: <36a5f74f.11285673@newshost.uwo.ca>
References: <77veih$11ts@b.stat.purdue.edu>
Newsgroups: sci.stat.math
Lines: 41

On 18 Jan 1999 08:56:33 -0500, hrubin@b.stat.purdue.edu (Herman Rubin) wrote:

>In article <36a23371.10418803@newshost.uwo.ca>,
>Duncan Murdoch <dmurdoch@pair.com> wrote:
>>On 17 Jan 1999 13:22:21 -0500, hrubin@b.stat.purdue.edu (Herman Rubin)
>>wrote:

>>On the other hand, if it's programmed in a way that is sometimes
>>incorrect in hard to detect ways, it might invalidate the whole
>>calculation. That seems like a much higher cost to pay.
>
>This is why it is important that those who understand the mathematics
>of the algorithm do it; I do not believe it pays to have a library
>routine written by one person. HLL algorithms fail in rather odd
>ways, and algorithms for generating random numbers, unless they have
>huge errors, can only be checked by direct proof of correctness.

Implementations of high-level languages often have bugs in them, but the language definitions are mostly specific about what will be calculated from a given input. That's also true about machine languages, but it's certainly easier to verify when the language is closer to the original mathematical specification.

>>>An idiot-proof language is only for idiots.
>
>>No, if such a thing existed, it would also reduce the occasional
>>lapses of non-idiots. Programmers are fallible; if tools can prevent
>>some common errors (even if it means doubling the computation time),
>>then those tools are worth using.
>
>On the contrary, these tools can cause the programmer to make
>errors which cannot be easily detected.

I'm not sure what you have in mind. What I had in mind are tools like strong type checking and optional range checking (which you don't get in machine languages). What sort of tools are you thinking of? Can you give examples where a tool designed to prevent errors actually introduces them?

Duncan Murdoch

Subject: Re: Random numbers for C: Improvements.
Date: 19 Jan 1999 11:27:03 -0500
From: hrubin@b.stat.purdue.edu (Herman Rubin)
Message-ID: <782bon$qjg@b.stat.purdue.edu>
References: <36a5f74f.11285673@newshost.uwo.ca>
Newsgroups: sci.stat.math
Lines: 121

In article <36a5f74f.11285673@newshost.uwo.ca>,
Duncan Murdoch <dmurdoch@pair.com> wrote:
>On 18 Jan 1999 08:56:33 -0500, hrubin@b.stat.purdue.edu (Herman Rubin)
>wrote:

>>In article <36a23371.10418803@newshost.uwo.ca>,
>>Duncan Murdoch <dmurdoch@pair.com> wrote:
>>>On 17 Jan 1999 13:22:21 -0500, hrubin@b.stat.purdue.edu (Herman Rubin)
>>>wrote:

>>>On the other hand, if it's programmed in a way that is sometimes
>>>incorrect in hard to detect ways, it might invalidate the whole
>>>calculation. That seems like a much higher cost to pay.

>>This is why it is important that those who understand the mathematics
>>of the algorithm do it; I do not believe it pays to have a library
>>routine written by one person. HLL algorithms fail in rather odd
>>ways, and algorithms for generating random numbers, unless they have
>>huge errors, can only be checked by direct proof of correctness.

>Implementations of high-level languages often have bugs in them, but
>the language definitions are mostly specific about what will be
>calculated from a given input. That's also true about machine
>languages, but it's certainly easier to verify when the language is
>closer to the original mathematical specification.

The problems are rather that the HLLs are not designed with what
the programmer can think of.

>>>>An idiot-proof language is only for idiots.

>>>No, if such a thing existed, it would also reduce the occasional
>>>lapses of non-idiots. Programmers are fallible; if tools can prevent
>>>some common errors (even if it means doubling the computation time),
>>>then those tools are worth using.

>>On the contrary, these tools can cause the programmer to make
>>errors which cannot be easily detected.

>I'm not sure what you have in mind. What I had in mind are tools like
>strong type checking and optional range checking (which you don't get
>in machine languages).

Can we afford range checking? And range checking could be added
to a machine language program; making it mandatory at least triples
the number of instructions needed for a memory reference, and one
of those is a conditional transfer. If we want it, it should be
put in as a hardware interrupt in a more complicated instruction,
where the efficiency loss will only be in the additional arguments.

Basic random tools should be as a bit stream, not as floating
numbers, for many reasons. If given a type, it should be integer,
or fixed point "real", not available in any of the common languages.
Floating point CANNOT do the job without major complications.
Efficient conversion is not available in most languages, and is
made worse with forced normalization. Without that, efficient
conversion is much easier, and multiple precision is much easier.
This hardware defect was promoted by the languages; the operations
needed did not occur in the languages, and thus those who looked
at the programs being written did not recognize that they were
considered useful.

Can we afford range checking? And range checking could be added
to a machine language program; making it mandatory at least triples
the number of instructions needed for a memory reference, and one
of those is a conditional transfer. If we want it, it should be
put in as a hardware interrupt in a more complicated instruction,
where the efficiency loss will only be in the additional arguments.

 What sort of tools are you thinking of?

Badly needed tools are more operations, not functions. The power
operation is a good example; even the possibility of using types
in C++ does not do the job. Good Fortran compilers expanded small
integers at compile time, and some even used this plus the square
root operation when the exponent was an integer plus .5. The C
power function always uses ln and exp. Also, pack and unpack are
needed, not what occurs in the languages. And even the frexp
function in C shows the inefficiency of not having a list of
results; the insistence of a store may more than double the cost
of using the operation.

One can have very complicated instructions, and still the hardware
efficiency of RISC, by putting much of the decoding in the unit
where the instruction is transferred. There were machines where
the first four bids were all that the instruction decoder looked
at, while certain other parts of the CPU looked at other bits
in parallel with this.

 Can
>you give examples where a tool designed to prevent errors actually
>introduces them?

Can we afford range checking? And range checking could be added
to a machine language program; making it mandatory at least triples
the number of instructions needed for a memory reference, and one
of those is a conditional transfer. If we want it, it should be
put in as a hardware interrupt in a more complicated instruction,
where the efficiency loss will only be in the additional arguments.

One good example of this occurs in calls to procedures generating
random numbers. As these calls do not have any arguments, or
the arguments are fixed locations, an "optimizing" compiler will
take them out of a loop!

The elaborate unnatural precedence structure in languages such as
C leads to errors. I have run into this with the precedence of
"+" and "<<" as "<<" is used for multiplication by a power of 2.

It was a real problem to get C to respect the users' parentheses.
Failure to do so can cause major loss of precision. Some modifications
of C allow binary floating point numbers; it is rare that one should
use a decimal floating point number in a subroutine. This has only
recently been introduced in C; it was possible to do this with an
illegal cast, but on little-endian machines, this was not easy.
--
This address is for information only. I do not claim that these views
are those of the Statistics Department or of Purdue University.
Herman Rubin, Dept. of Statistics, Purdue Univ., West Lafayette IN47907-1399
hrubin@stat.purdue.edu   Phone: (765)494-6054   FAX: (765)494-0558

Subject: Re: Random numbers for C: Improvements.
Date: Wed, 20 Jan 1999 02:31:55 GMT
From: dmurdoch@pair.com (Duncan Murdoch)
Message-ID: <36a63ccd.18681450@newshost.uwo.ca>
References: <782bon$qjg@b.stat.purdue.edu>
Newsgroups: sci.stat.math
Lines: 43

On 19 Jan 1999 11:27:03 -0500, hrubin@b.stat.purdue.edu (Herman Rubin)
wrote:

>>I'm not sure what you have in mind. What I had in mind are tools like
>>strong type checking and optional range checking (which you don't get
>>in machine languages).
>
>Can we afford range checking? And range checking could be added
>to a machine language program; making it mandatory at least triples
>the number of instructions needed for a memory reference, and one
>of those is a conditional transfer. If we want it, it should be
>put in as a hardware interrupt in a more complicated instruction,
>where the efficiency loss will only be in the additional arguments.

I think Java is the only language I've come across that makes range
checking mandatory. Most implementations make it optional: you
supply the same source code, the compiler produces different machine
code depending on how you set the option. That could possibly be done
in an assembler, but I've never heard of anyone trying it.

The normal use for range checking is in debugging. If you have to
change your source code between debugging and running, there's a big
chance of introducing new bugs.

>The elaborate unnatural precedence structure in languages such as
>C leads to errors. I have run into this with the precedence of
>"+" and "<<" as "<<" is used for multiplication by a power of 2.

Someone else will have to argue for C. I think it's hardly better
than assembler, myself.

>It was a real problem to get C to respect the users' parentheses.

I think it's pretty common for compilers to rearrange expressions to
make them more convenient to calculate, under the incorrect assumption
that mathematically equivalent expressions will generate equal
results. But all languages that I know (including C) provide ways of
forcing the order of evaluation: just put the things that need to be
done first in an earlier statement. This is exactly the way you do it
in assembler.

Duncan Murdoch

Subject: Re: Random numbers for C: Improvements.
Date: Sun, 17 Jan 1999 14:33:32 -0500
From: dwnoon@compuserve.com (David W. Noon)
Message-ID: <cUwZzZKf1ety-pn2-vB2YCWNn5fgR@mfs-pci-bqh-vty163.as.wcom.net>
References: <77t9ot$15ha@b.stat.purdue.edu>
Newsgroups: sci.math.num-analysis,sci.stat.math,sci.math
Lines: 62

On Sun, 17 Jan 1999 18:22:21, hrubin@b.stat.purdue.edu (Herman Rubin)
wrote:

> In article <99Jan15.151547edt.785@neuron.ai.toronto.edu>,
> Radford Neal <radford@cs.toronto.edu> wrote:
> >>George Marsaglia <geo@stat.fsu.edu> writes:
>
> >>> #define LFIB4 (t[c]=t[c]+t[c+58]+t[c+119]+t[c+178],t[++c])
> >>> #define SWB   (t[c+237]=(x=t[c+15])-(y=t[c]+(x<y)),t[++c])
>
>
> >In article <790f49x60.fsf@poole.statgen.ncsu.edu>,
> >Dmitri Zaykin <zaykin@statgen.ncsu.edu> wrote:
>
> >>Shouldn't these be
>
> >>#define LFIB4 (t[c]=t[c]+t[c+58]+t[c+119]+t[c+179], ++c)
> >>#define SWB   (t[c+237]=(x=t[c+15])-(y=t[c+1]+(x<y)), ++c)
>
>
> >This doesn't work either. I believe that it is undefined whether the
> >comparison x<y uses the new or the old value of x. The SHR3 macro
> >in the original source also suffers from this flaw.
>
> >I think one needs to face up to an unpleasant truth: The #define
> >facility of C was poorly designed, and is incapable in general of
> >supporting the definition of "in-line" procedures. It is far better
> >to simply write ordinary C procedures, and accept the fairly small
> >procedure call overhead.
>
> I think it should be clarified, and probably written out in some
> more detail. But the procedure call overhead would be comparable
> to the computing cost; C, and all other languages, have such great
> built-in inefficiencies that what is needed is something written
> from the standpoint of mathematics and efficiency.
>
> But even the use of a comparison is a costly operation, if the
> result is to be used promptly. Any form of random, pseudo-random,
> or quasi-random numbers should be done in good integer form,
> not present now on many machines because of bad design coming
> from bad languages which have thrown out the natural properties
> of computers, and attempts to keep stupid people from making
> mistakes. An idiot-proof language is only for idiots.

This is very admirable, but it is basically saying that the code
should be in assembler. There goes portability.

I don't mind translating all this stuff into assembler, but anybody
not using an Intel 80486 or better running a 32-bit version of OS/2
will not be able to use my code.

The only other language I know of that has sufficiently powerful macro
facilities to force inlining like C but with better syntax, is
portable across many platforms, and produces efficient object code is
PL/I. If anybody besides me can use such code, I will produce it
too/instead.

So, who wants it?

Regards

Dave

Subject: Re: Random numbers for C: Improvements.
Date: 18 Jan 1999 00:16:24 GMT
From: zaykin@statgen.ncsu.edu (Dmitri Zaykin)
Message-ID: <5tFXN12Xwb3.7WA43126hM3916@statgen.ncsu.edu>
References: <cUwZzZKf1ety-pn2-vB2YCWNn5fgR@mfs-pci-bqh-vty163.as.wcom.net>
Newsgroups: sci.math.num-analysis,sci.stat.math,sci.math
Lines: 78

dwnoon@compuserve.com (David W. Noon) writes:
> The only other language I know of that has sufficiently powerful macro
> facilities to force inlining like C but with better syntax, is
> portable across many platforms, and produces efficient object code is
> PL/I. If anybody besides me can use such code, I will produce it
> too/instead.

Another option is to re-write macros as C++ member functions. If all
the code is in the body of the class or "inline" keyword is explicitly
given, the compiler will attempt to inline the functions.

Below is my version of it and an example of usage. I've put "//"
comments in places where I thought changes to the C code are
necessary. There are additional advantages over C-macros. (1) No

global variables are introduced. (2) It is easy to run several
"independent" random sequences by creating several variables of Rnd
type and seeding them differently (something that would not be
straightforward to do in C).

Dmitri

```
#include &LT;limits.h> // ULONG_MAX and UCHAR_MAX defined there

class Rnd {
    Rnd() {}
    typedef unsigned long Unlo;
    Unlo z, w, jsr, jcong, t[UCHAR_MAX+1], x, y;
    unsigned char c;
    Unlo znew() { return (z = 36969UL*(z & 65535UL)+(z >> 16)); } // +UL
    Unlo wnew() { return (w = 18000UL*(w & 65535UL)+(w >> 16)); } // +UL
 public:
    Rnd (Unlo i1, Unlo i2, Unlo i3, Unlo i4)
            : z(i1), w(i2), jsr(i3), jcong(i4), x(0), y(0), c(0) {
        for(int i=0; i&LT;UCHAR_MAX+1; i++) t[i] = Kiss();
    }
    Unlo Mwc() { return (znew() &LT;&LT; 16) + wnew(); }
    Unlo Shr3 () { // was:
jsr=(jsr=(jsr=jsr^(jsr&LT;&LT;17))^(jsr>>13))^(jsr&LT;&LT;5)
        jsr=jsr^(jsr&LT;&LT;17);
        jsr=jsr^(jsr>>13);
        return (jsr=jsr^(jsr&LT;&LT;5));
    }
    Unlo Cong() { return (jcong = 69069UL*jcong + 1234567UL); } // +UL
    Unlo Kiss() { return (Mwc() ^ Cong()) + Shr3(); }
    Unlo Swb () { // was: t[c+237]=(x=t[c+15])-(y=t[c]+(x&LT;y)),t[++c]
        x = t[(c+15) & 0xFF];
        t[(c+237) & 0xFF] = x - (y = t[(c+1) & 0xFF] + (x &LT; y));
        return t[++c];
    }
    Unlo Lfib4() { // was: t[c]=t[c]+t[c+58]+t[c+119]+t[c+178],t[++c]
        t[c]=t[c]+t[(c+58) & 0xFF]+t[(c+119) & 0xFF]+t[(c+179) & 0xFF];
        return t[++c];
    }
    double Uni() { return Kiss() * 2.328306e-10; }
    double Vni() { return long(Kiss()) * 4.656613e-10; }
    double operator () () { return Uni(); }
    Unlo operator () (Unlo n) {
        return n == 1 ? 0 : Kiss() / (ULONG_MAX/n + 1);
    }
    double operator () (double Min, double Max) { return Min+Uni()*(Max-Min); }
};

// example of usage

#include &LT;time.h>
#include &LT;iostream.h>

int main()
{
    unsigned i, seed=time(0);
    Rnd rn (seed, 2*seed, 3*seed, 4*seed);

    for(i=0; i&LT;5; i++) cout &LT;&LT; rn(5) &LT;&LT; endl;      // [0, 1, 2, 3, 4]
    for(i=0; i&LT;5; i++) cout &LT;&LT; rn() &LT;&LT; endl;       // (0, ..., 1)
    for(i=0; i&LT;5; i++) cout &LT;&LT; rn.Vni() &LT;&LT; endl;   // (-1, ..., 1)
    for(i=0; i&LT;5; i++) cout &LT;&LT; rn(10, 20) &LT;&LT; endl; // (10, ..., 20)
    for(i=0; i&LT;5; i++) cout &LT;&LT; rn.Lfib4() &LT;&LT; endl; // trying Lfib4
    for(i=0; i&LT;5; i++) cout &LT;&LT; rn.Swb() &LT;&LT; endl;   // trying Swb
}
```

Subject: Re: Random numbers for C: Improvements.
Date: Mon, 18 Jan 1999 17:07:26 -0500
From: dwnoon@compuserve.com (David W. Noon)
Message-ID: <cUwZzZKf1ety-pn2-NxnsgFONxOGO@localhost>
References: <5tFXN12Xwb3.7WA43126hM3916@statgen.ncsu.edu>
Newsgroups: sci.stat.math,sci.math,sci.math.num-analysis
Lines: 97

On Mon, 18 Jan 1999 00:16:24, zaykin@statgen.ncsu.edu (Dmitri Zaykin)
wrote:

> dwnoon@compuserve.com (David W. Noon) writes:
> > The only other language I know of that has sufficiently powerful macro
> > facilities to force inlining like C but with better syntax, is
> > portable across many platforms, and produces efficient object code is
> > PL/I. If anybody besides me can use such code, I will produce it
> > too/instead.
>
> Another option is to re-write macros as C++ member functions. If all
> the code is in the body of the class or "inline" keyword is explicitly
> given, the compiler will attempt to inline the functions.
>
> Below is my version of it and an example of usage. I've put "//"
> comments in places where I thought changes to the C code are
> necessary. There are additional advantages over C-macros. (1) No
> global variables are introduced. (2) It is easy to run several
> "independent" random sequences by creating several variables of Rnd

```
> type and seeding them differently (something that would not be
> straightforward to do in C).
>
> Dmitri
>
> #include &LT;limits.h> // ULONG_MAX and UCHAR_MAX defined there
>
> class Rnd {
>     Rnd() {}
>     typedef unsigned long Unlo;
>     Unlo z, w, jsr, jcong, t[UCHAR_MAX+1], x, y;
>     unsigned char c;
>     Unlo znew() { return (z = 36969UL*(z & 65535UL)+(z >> 16)); } // +UL
>     Unlo wnew() { return (w = 18000UL*(w & 65535UL)+(w >> 16)); } // +UL
>  public:
>     Rnd (Unlo i1, Unlo i2, Unlo i3, Unlo i4)
>             : z(i1), w(i2), jsr(i3), jcong(i4), x(0), y(0), c(0) {
>         for(int i=0; i&LT;UCHAR_MAX+1; i++) t[i] = Kiss();
>     }
>     Unlo Mwc() { return (znew() &LT;&LT; 16) + wnew(); }
>     Unlo Shr3 () { // was:
jsr=(jsr=(jsr=jsr^(jsr&LT;&LT;17))^(jsr>>13))^(jsr&LT;&LT;5)
>         jsr=jsr^(jsr&LT;&LT;17);
>         jsr=jsr^(jsr>>13);
>         return (jsr=jsr^(jsr&LT;&LT;5));
>     }
>     Unlo Cong() { return (jcong = 69069UL*jcong + 1234567UL); } // +UL
>     Unlo Kiss() { return (Mwc() ^ Cong()) + Shr3(); }
>     Unlo Swb () { // was: t[c+237]=(x=t[c+15])-(y=t[c]+(x&LT;y)),t[++c]
>         x = t[(c+15) & 0xFF];
>         t[(c+237) & 0xFF] = x - (y = t[(c+1) & 0xFF] + (x &LT; y));
>         return t[++c];
>     }
>     Unlo Lfib4() { // was: t[c]=t[c]+t[c+58]+t[c+119]+t[c+178],t[++c]
>         t[c]=t[c]+t[(c+58) & 0xFF]+t[(c+119) & 0xFF]+t[(c+179) & 0xFF];
>         return t[++c];
>     }
>     double Uni() { return Kiss() * 2.328306e-10; }
>     double Vni() { return long(Kiss()) * 4.656613e-10; }
>     double operator () () { return Uni(); }
>     Unlo operator () (Unlo n) {
>         return n == 1 ? 0 : Kiss() / (ULONG_MAX/n + 1);
>     }
>     double operator () (double Min, double Max) { return Min+Uni()*(Max-Min); }
> };
>
> // example of usage
>
> #include &LT;time.h>
> #include &LT;iostream.h>
>
> int main()
> {
>     unsigned i, seed=time(0);
>     Rnd rn (seed, 2*seed, 3*seed, 4*seed);
>
>     for(i=0; i&LT;5; i++) cout &LT;&LT; rn(5) &LT;&LT; endl;      // [0, 1, 2, 3,
4]
>     for(i=0; i&LT;5; i++) cout &LT;&LT; rn() &LT;&LT; endl;       // (0, ..., 1)
>     for(i=0; i&LT;5; i++) cout &LT;&LT; rn.Vni() &LT;&LT; endl;   // (-1, ..., 1)
>     for(i=0; i&LT;5; i++) cout &LT;&LT; rn(10, 20) &LT;&LT; endl; // (10, ..., 20)
>     for(i=0; i&LT;5; i++) cout &LT;&LT; rn.Lfib4() &LT;&LT; endl; // trying Lfib4
>     for(i=0; i&LT;5; i++) cout &LT;&LT; rn.Swb() &LT;&LT; endl;   // trying Swb
> }
```

The problem with C++ is that VMT calling mechanisms are almost invariably slower than the calling mechanisms used by C, PL/I, FORTRAN, Pascal, etc. [You have an extra parameter (this) to pass, even if you don't use it, and you have to do a VMT lookup to find the method.] So, while you do have a way to inline with some elegance, it can be a chore to get to the inlined code from the main program.

If you want to use C++, use a template instead of a class. It's much faster, usually, but still usually slower than C, PL/I or FORTRAN -- and _way_ slower than assembler. For PRNG speed _is_ important, as PRN's are usually generated in bulk.

Regards

Dave

Subject: Re: Random numbers for C: Improvements.
Date: 18 Jan 1999 17:57:49 -0500
From: zaykin@statgen.ncsu.edu (Dmitri Zaykin)
Message-ID: <7sod842aa.fsf@poole.statgen.ncsu.edu>
References: <cUwZzZKf1ety-pn2-NxnsgFONxOGO@localhost>
Newsgroups: sci.stat.math,sci.math,sci.math.num-analysis
Lines: 18

dwnoon@compuserve.com (David W. Noon) writes:
> The problem with C++ is that VMT calling mechanisms are almost
> invariably slower than the calling mechanisms used by C, PL/I,
> FORTRAN, Pascal, etc. [You have an extra parameter (this) to pass,

> even if you don't use it, and you have to do a VMT lookup to find the
> method.]

Well, in this particular case all that does not apply, since there are no virtual functions in the code. As C++ FAQ says, "The compiler creates a v-table for each class that has at least one virtual function."

(http://www.cerfnet.com/~mpcline/c++-faq-lite/virtual-functions.html)

Regular member functions and overloaded operators are resolved at compile time (in this case, inlined), so it gets as good as C-macros.

Dmitri

Subject: Re: Random numbers for C: Improvements.
Date: Tue, 19 Jan 1999 03:39:07 GMT
From: dmurdoch@pair.com (Duncan Murdoch)
Message-ID: <36a3fc77.12606540@newshost.uwo.ca>
References: <cUwZzZKf1ety-pn2-NxnsgFONxOGO@localhost>
Newsgroups: sci.stat.math
Lines: 20

On Mon, 18 Jan 1999 17:07:26 -0500, dwnoon@compuserve.com (David W. Noon) wrote:

>If you want to use C++, use a template instead of a class. It's much
>faster, usually, but still usually slower than C, PL/I or FORTRAN --
>and _way_ slower than assembler. For PRNG speed _is_ important, as
>PRN's are usually generated in bulk.

...but just as with any other aspect of the program, speed isn't as important as correctness.

If I can write tricky routines in assembler that are 10 times faster than clear ones, then in a given length of computational time my Monte Carlo integration will get an extra half digit of precision.

On the other hand, if I mess up the implementation because I was just a little bit too clever for my own good, it might be that none of the digits are right.

Duncan Murdoch

Subject: Re: Random numbers for C: Improvements.
Date: 20 Jan 1999 21:32:39 -0500
From: zaykin@statgen.ncsu.edu (Dmitri Zaykin)
Message-ID: <7sod5s6d4.fsf@poole.statgen.ncsu.edu>
References: <cUwZzZKf1ety-pn2-NxnsgFONxOGO@localhost>
Newsgroups: sci.stat.math,sci.math,sci.math.num-analysis
Lines: 131

To check if C-macros for these random number generators do indeed always produce faster, and maybe "less bloated" code than inlined C++ member functions, I did a little experiment with timing/code size using the GNU compiler (egcs-1.1.1 g++/gcc) on Solaris 2.5.1. With this compiler, it is clearly not the case.

(1) Code size (conclusion: C++ code smaller)

inlined member functions g++ -Winline -O2 -s	C-macros g++ -Winline -O2 -s	C-macros gcc -Winline -O2 -s
8908	9820	9532

(2) Timing in 10 experiments (conclusion: C++ code faster)

inlined member functions g++ -Winline -O2 -s	C-macros g++ -Winline -O2 -s	C-macros gcc -Winline -O2 -s
11330000	15030000	14500000
11330000	15040000	14470000
11340000	15030000	14490000
11340000	15040000	14520000
11340000	15030000	14500000
11320000	15030000	14490000
11340000	15040000	14480000
11340000	15030000	14510000
11340000	15030000	14500000
11340000	15030000	14500000

```
//-----------------------------------------------------
// C++ code:

#include &LT;stdio.h>
#include &LT;time.h>
#include &LT;limits.h>

class Rnd {
    Rnd() {}
    typedef unsigned long Unlo;
    typedef unsigned char Uc;
    Unlo z, w, jsr, jcong, t[UCHAR_MAX+1], x, y, a, b;
```

```
   unsigned char c;
   Unlo znew() { return (z = 36969UL*(z & 65535UL)+(z >> 16)); }
   Unlo wnew() { return (w = 18000UL*(w & 65535UL)+(w >> 16)); }
public:
   Rnd (Unlo i1, Unlo i2, Unlo i3, Unlo i4, Unlo i5, Unlo i6)
           : z(i1), w(i2), jsr(i3), jcong(i4), x(0), y(0),
             a(i5), b(i6), c(0) {
       for(int i=0; i&LT;UCHAR_MAX+1; i++) t[i] = Kiss();
   }
   Unlo Mwc() { return (znew() &LT;&LT; 16) + wnew(); }
   Unlo Shr3 () {
       jsr=jsr^(jsr&LT;&LT;17);
       jsr=jsr^(jsr>>13);
       return (jsr=jsr^(jsr&LT;&LT;5));
   }
   Unlo Cong() { return (jcong = 69069UL*jcong + 1234567UL); }
   Unlo Kiss() { return (Mwc() ^ Cong()) + Shr3(); }
   Unlo Swb () {
       x = t[(Uc)(c+15)];
       t[(Uc)(c+237)] = x - (y = t[(Uc)(c+1)] + (x &LT; y));
       return t[++c];
   }
   Unlo Lfib4() {
       t[c]=t[c]+t[(Uc)(c+58)]+t[(Uc)(c+119)]+t[(Uc)(c+179)];
       return t[++c];
   }
   Unlo Fib() { b=a+b; return (a=b-a); }
   double Uni() { return Kiss() * 2.328306e-10; }
   double Vni() { return long(Kiss()) * 4.656613e-10; }
   double operator () () { return Uni(); }
   Unlo operator () (Unlo n) {
       return n == 1 ? 0 : Kiss() / (ULONG_MAX/n + 1);
   }
   double operator () (double Min, double Max) { return Min+Uni()*(Max-Min); }
};

int main()
{
   unsigned long i, xx=0, seed=time(0);
   long spent;
   Rnd rn (seed, 2*seed, 3*seed, 4*seed, 5*seed, 6*seed);

   spent=clock();
   for(i=0; i&LT;77777777; i++) xx += (rn.Kiss() + rn.Swb());
   printf ("%ld \t", clock()-spent);

   printf("\n");
}

/*************************************************************/
/* C-macros */

#include &LT;stdio.h>
#include &LT;time.h>
#include &LT;limits.h>

#define znew   (z=36969UL*(z&65535UL)+(z>>16))
#define wnew   (w=18000UL*(w&65535UL)+(w>>16))
#define MWC    ((znew&LT;&LT;16)+wnew )
#define SHR3  (jsr^=(jsr&LT;&LT;17), jsr^=(jsr>>13), jsr^=(jsr&LT;&LT;5))
#define CONG  (jcong=69069UL*jcong+1234567UL)
#define FIB   ((b=a+b),(a=b-a))
#define KISS  ((MWC^CONG)+SHR3)
#define UC    (unsigned char)
#define LFIB4 (c++,t[c]=t[c]+t[UC(c+58)]+t[UC(c+119)]+t[UC(c+178)])
#define SWB   (x = t[UC(c+15)], t[UC(c+237)] = x-(y=t[UC(c+1)]+(x&LT;y)), t[++c])
#define UNI   (KISS*2.328306e-10)
#define VNI   ((long) KISS)*4.656613e-10
typedef unsigned long Un;
static Un z=362436069UL, w=521288629UL, jsr=123456789UL, jcong=380116160UL;
static Un a=224466889UL, b=7584631UL, t[256];
static Un x=0,y=0; static unsigned char c=0;
void settable(Un i1,Un i2,Un i3,Un i4,Un i5, Un i6)
{ int i; z=i1;w=i2,jsr=i3; jcong=i4; a=i5; b=i6;
 for(i=0;i&LT;256;i=i+1)  t[i]=KISS;
}

int main()
{
   unsigned long i, xx=0, seed=time(0);
   long spent;

   settable (seed, 2*seed, 3*seed, 4*seed, 5*seed, 6*seed);

   spent=clock();
   for(i=0; i&LT;77777777; i++) xx += (KISS + SWB);
   printf ("%ld \t", spent=clock()-spent);

   printf("\n");
   return 0;
}
```

Subject: Re: Random numbers for C: Improvements.
Date: 21 Jan 1999 04:40:53 GMT

From: zaykin@statgen.ncsu.edu (Dmitri Zaykin)
Message-ID: <124123g26m11754.1356cjx442854@statgen.ncsu.edu>
References: <cUwZzZKf1ety-pn2-NxnsgFONxOGO@localhost>
Newsgroups: sci.stat.math,sci.math,sci.math.num-analysis
Lines: 131

To check if C-macros for these random number generators do indeed always produce faster, and maybe "less bloated" code than inlined C++ member functions, I did a little experiment with timing/code size using the GNU compiler (egcs-1.1.1 g++/gcc) on Solaris 2.5.1. With this compiler, it is clearly not the case.

(1) Executable size (conclusion: C++ code smaller)

inlined member functions	C-macros	C-macros
g++ -Winline -O2 -s	g++ -Winline -O2 -s	gcc -Winline -O2 -s
8908	9820	9532

(2) Timing in 10 experiments (conclusion: C++ code faster)

inlined member functions	C-macros	C-macros
g++ -Winline -O2 -s	g++ -Winline -O2 -s	gcc -Winline -O2 -s
11330000	15030000	14500000
11330000	15040000	14470000
11340000	15030000	14490000
11340000	15040000	14520000
11340000	15030000	14500000
11320000	15030000	14490000
11340000	15040000	14480000
11340000	15030000	14510000
11340000	15030000	14500000
11340000	15030000	14500000

```
//-----------------------------------------------------
// C++ code:

#include &LT;stdio.h>
#include &LT;time.h>
#include &LT;limits.h>

class Rnd {
    Rnd() {}
    typedef unsigned long Unlo;
    typedef unsigned char Uc;
    Unlo z, w, jsr, jcong, t[UCHAR_MAX+1], x, y, a, b;
    unsigned char c;
    Unlo znew() { return (z = 36969UL*(z & 65535UL)+(z >> 16)); }
    Unlo wnew() { return (w = 18000UL*(w & 65535UL)+(w >> 16)); }
 public:
    Rnd (Unlo i1, Unlo i2, Unlo i3, Unlo i4, Unlo i5, Unlo i6)
            : z(i1), w(i2), jsr(i3), jcong(i4), x(0), y(0),
              a(i5), b(i6), c(0) {
        for(int i=0; i&LT;UCHAR_MAX+1; i++) t[i] = Kiss();
    }
    Unlo Mwc() { return (znew() &LT;&LT; 16) + wnew(); }
    Unlo Shr3 () {
        jsr=jsr^(jsr&LT;&LT;17);
        jsr=jsr^(jsr>>13);
        return (jsr=jsr^(jsr&LT;&LT;5));
    }
    Unlo Cong() { return (jcong = 69069UL*jcong + 1234567UL); }
    Unlo Kiss() { return (Mwc() ^ Cong()) + Shr3(); }
    Unlo Swb () {
        x = t[(Uc)(c+15)];
        t[(Uc)(c+237)] = x - (y = t[(Uc)(c+1)] + (x &LT; y));
        return t[++c];
    }
    Unlo Lfib4() {
        t[c]=t[c]+t[(Uc)(c+58)]+t[(Uc)(c+119)]+t[(Uc)(c+179)];
        return t[++c];
    }
    Unlo Fib() { b=a+b; return (a=b-a); }
    double Uni() { return Kiss() * 2.328306e-10; }
    double Vni() { return long(Kiss()) * 4.656613e-10; }
    double operator () () { return Uni(); }
    Unlo operator () (Unlo n) {
        return n == 1 ? 0 : Kiss() / (ULONG_MAX/n + 1);
    }
    double operator () (double Min, double Max) { return Min+Uni()*(Max-Min); }
};

int main()
{
    unsigned long i, xx=0, seed=time(0);
    long spent;
    Rnd rn (seed, 2*seed, 3*seed, 4*seed, 5*seed, 6*seed);

    spent=clock();
    for(i=0; i&LT;77777777; i++) xx += (rn.Kiss() + rn.Swb());
    printf ("%ld \t", clock()-spent);

    printf("\n");
}
```

```
/*****************************************************************/
/* C-macros */

#include &LT;stdio.h>
#include &LT;time.h>
#include &LT;limits.h>

#define znew   (z=36969UL*(z&65535UL)+(z>>16))
#define wnew   (w=18000UL*(w&65535UL)+(w>>16))
#define MWC    ((znew&LT;&LT;16)+wnew )
#define SHR3  (jsr^=(jsr&LT;&LT;17), jsr^=(jsr>>13), jsr^=(jsr&LT;&LT;5))
#define CONG  (jcong=69069UL*jcong+1234567UL)
#define FIB   ((b=a+b),(a=b-a))
#define KISS  ((MWC^CONG)+SHR3)
#define UC    (unsigned char)
#define LFIB4 (c++,t[c]=t[c]+t[UC(c+58)]+t[UC(c+119)]+t[UC(c+178)])
#define SWB   (x = t[UC(c+15)], t[UC(c+237)] = x-(y=t[UC(c+1)]+(x&LT;y)), t[++c])
#define UNI   (KISS*2.328306e-10)
#define VNI   ((long) KISS)*4.656613e-10
typedef unsigned long Un;
static Un z=362436069UL, w=521288629UL, jsr=123456789UL, jcong=380116160UL;
static Un a=224466889UL, b=7584631UL, t[256];
static Un x=0,y=0; static unsigned char c=0;
void settable(Un i1,Un i2,Un i3,Un i4,Un i5, Un i6)
{ int i; z=i1;w=i2,jsr=i3; jcong=i4; a=i5; b=i6;
 for(i=0;i&LT;256;i=i+1)  t[i]=KISS;
}

int main()
{
    unsigned long i, xx=0, seed=time(0);
    long spent;

    settable (seed, 2*seed, 3*seed, 4*seed, 5*seed, 6*seed);

    spent=clock();
    for(i=0; i&LT;77777777; i++) xx += (KISS + SWB);
    printf ("%ld \t", clock()-spent);

    printf("\n");
    return 0;
}

User-Agent: slrn/0.9.5.4 (UNIX)
```

Subject: Re: Random numbers for C: Improvements.
Date: 21 Jan 1999 18:01:49 GMT
From: davis@space.mit.edu (John E. Davis)
Message-ID: <slrn7aeqsa.bkc.davis@aluche.mit.edu>
References: <124123g26m11754.1356cjx442854@statgen.ncsu.edu>
Newsgroups: sci.stat.math,sci.math,sci.math.num-analysis
Lines: 38

On 21 Jan 1999 04:40:53 GMT, Dmitri Zaykin <zaykin@statgen.ncsu.edu>
wrote:
>(1) Executable size (conclusion: C++ code smaller)
[...]
>(2) Timing in 10 experiments (conclusion: C++ code faster)

Using your code (t.cc and t.c), my conclusion is the opposite:

```
Script started on Thu Jan 21 12:54:02 1999
$ gcc -Winline -O2 -s t.c
$ ls -l a.out
-rwxr-xr-x   1 davis    asc         8752 Jan 21 12:54 a.out*
$ ./a.out
21700000
$ g++ -Winline -O2 -s t.cc
t.cc: In method `Rnd::Rnd(long unsigned int, long unsigned int, long unsigned int,
long unsigned int, long unsigned int, long unsigned int)':
t.cc:29: warning: can't inline call to `long unsigned int Rnd::Kiss()'
t.cc:20: warning: called from here
$ ls -l a.out
-rwxr-xr-x   1 davis    asc         8880 Jan 21 12:55 a.out*
$ ./a.out
25250000
$ uname -a
SunOS wiwaxia 5.6 Generic_105181-08 sun4u sparc SUNW,Ultra-2
$ gcc -v
Reading specs from /usr/local/lib/gcc-lib/sparc-sun-solaris2.6/2.8.1/specs
gcc version 2.8.1
$ g++ -v
Reading specs from /usr/local/lib/gcc-lib/sparc-sun-solaris2.6/2.8.1/specs
gcc version 2.8.1
$ exit
exit

script done on Thu Jan 21 12:56:40 1999
```

--
John E. Davis Center for Space Research/AXAF Science Center
617-258-8119 One Hampshire St., Building NE80-6019
http://space.mit.edu/~davis Cambridge, MA 02139-4307

Subject: Re: Random numbers for C: Improvements.
Date: 21 Jan 1999 16:19:55 -0500
From: zaykin@statgen.ncsu.edu (Dmitri Zaykin)
Message-ID: <7btjsuxvo.fsf@poole.statgen.ncsu.edu>
References: <slrn7aeqsa.bkc.davis@aluche.mit.edu>
Newsgroups: sci.stat.math,sci.math,sci.math.num-analysis
Lines: 11

davis@space.mit.edu (John E. Davis) writes:
> Using your code (t.cc and t.c), my conclusion is the opposite:
>
> Reading specs from /usr/local/lib/gcc-lib/sparc-sun-solaris2.6/2.8.1/specs
> gcc version 2.8.1

I used egcs-1.1.1 compiler. It is is an improvement over gcc-2.8.1

http://egcs.cygnus.com/egcs-1.1/egcs-1.1.1.html

Dmitri

Subject: Re: Random numbers for C: Improvements.
Date: 18 Jan 1999 21:12:35 GMT
From: jmccarty@sun1307.spd.dsccc.com (Mike McCarty)
Message-ID: <780843ikj1@relay1.dsccc.com>
References: <cUwZzZKf1ety-pn2-vB2YCWNn5fgR@mfs-pci-bqh-vty163.as.wcom.net>
Newsgroups: sci.math.num-analysis,sci.stat.math,sci.math
Lines: 23

In article <cUwZzZKf1ety-pn2-vB2YCWNn5fgR@mfs-pci-bqh-vty163.as.wcom.net>,
David W. Noon <dwnoon@compuserve.com> wrote:

[snip]

)This is very admirable, but it is basically saying that the code
)should be in assembler. There goes portability.
)
)I don't mind translating all this stuff into assembler, but anybody
)not using an Intel 80486 or better running a 32-bit version of OS/2
)will not be able to use my code.
)
)The only other language I know of that has sufficiently powerful macro
)facilities to force inlining like C but with better syntax, is
)portable across many platforms, and produces efficient object code is
)PL/I. If anybody besides me can use such code, I will produce it

You haven't heard of C++?
--
----
char *p="char *p=%c%s%c;main(){printf(p,34,p,34);}";main(){printf(p,34,p,34);}
This message made from 100% recycled bits.
I don't speak for Alcatel <- They make me say that.

Subject: Re: Random numbers for C: Improvements.
Date: 18 Jan 1999 19:27:13 -0500
From: hrubin@odds.stat.purdue.edu (Herman Rubin)
Message-ID: <780jh1$11ii@odds.stat.purdue.edu>
References: <780843ikj1@relay1.dsccc.com>
Newsgroups: sci.math.num-analysis,sci.stat.math,sci.math
Lines: 45

In article <780843ikj1@relay1.dsccc.com>,
Mike McCarty <jmccarty@sun1307.spd.dsccc.com> wrote:
>In article <cUwZzZKf1ety-pn2-vB2YCWNn5fgR@mfs-pci-bqh-vty163.as.wcom.net>,
>David W. Noon <dwnoon@compuserve.com> wrote:

[snip]

>)This is very admirable, but it is basically saying that the code
>)should be in assembler. There goes portability.

Decent random number code cannot be portable. Anything which involves
bit handling, as this should, suffers from the problem. And besides
the problem of portability of code, there is the portability of results.
The order of generating two non-uniform random numbers, or the change
of size of a buffer, can do this with ease.

>)I don't mind translating all this stuff into assembler, but anybody
>)not using an Intel 80486 or better running a 32-bit version of OS/2
>)will not be able to use my code.

>)The only other language I know of that has sufficiently powerful macro
>)facilities to force inlining like C but with better syntax, is
>)portable across many platforms, and produces efficient object code is
>)PL/I. If anybody besides me can use such code, I will produce it

I doubt that PL/I can do the job. But I would welcome trying it.

Let me give you an important tool; this is to generate the distance
to the next bit in a stream of random bits, move the pointer, and
interrupt if the stream becomes empty. This is a TOOL; for some
purposes, it will be used more frequently than additions.

To illustrate what one can do with it, suppose we want to generate
random variables with density 4*x*(1-x) on (0,1). Generate two
random variables A and B as above. Let K be A/2, rounded up, and
let N be K+B. Then change the N-th bit to the right of the binary
point of a uniform (0,1) random variable X to the opposite of the
K-th to get the result. BTW, I believe you can see that to do this
with a floating point representation of X is quite difficult.

--
This address is for information only. I do not claim that these views
are those of the Statistics Department or of Purdue University.
Herman Rubin, Dept. of Statistics, Purdue Univ., West Lafayette IN47907-1399
hrubin@stat.purdue.edu Phone: (765)494-6054 FAX: (765)494-0558

Subject: Re: Random numbers for C: Improvements.
Date: Tue, 19 Jan 1999 18:09:39 -0500
From: dwnoon@compuserve.com (David W. Noon)
Message-ID: <cUwZzZKf1ety-pn2-deRYWonQPOXL@localhost>
References: <780jh1$11ii@odds.stat.purdue.edu>
Newsgroups: sci.stat.math,sci.math,sci.math.num-analysis
Lines: 86

On Tue, 19 Jan 1999 00:27:13, hrubin@odds.stat.purdue.edu (Herman
Rubin) wrote:

> >In article <cUwZzZKf1ety-pn2-vB2YCWNn5fgR@mfs-pci-bqh-vty163.as.wcom.net>,
> >David W. Noon <dwnoon@compuserve.com> wrote:
>
> [snip]
>
> >)This is very admirable, but it is basically saying that the code
> >)should be in assembler. There goes portability.
>
> Decent random number code cannot be portable. Anything which involves
> bit handling, as this should, suffers from the problem. And besides
> the problem of portability of code, there is the portability of results.
> The order of generating two non-uniform random numbers, or the change
> of size of a buffer, can do this with ease.

I meant portable across software platforms, not hardware.

I would hope your code executes on any 32-bit Intel system the same
way, whether that system is running Windows NT, OS/2, Linux, any UNIX
variant or whatever.

> >)I don't mind translating all this stuff into assembler, but anybody
> >)not using an Intel 80486 or better running a 32-bit version of OS/2
> >)will not be able to use my code.
>
> >)The only other language I know of that has sufficiently powerful macro
> >)facilities to force inlining like C but with better syntax, is
> >)portable across many platforms, and produces efficient object code is
> >)PL/I. If anybody besides me can use such code, I will produce it
>
> I doubt that PL/I can do the job. But I would welcome trying it.

It can certainly handle the code that you have posted so far, and with
ease.

> Let me give you an important tool; this is to generate the distance
> to the next bit in a stream of random bits,

Surely the distance to the next bit is always 1 position. Did you mean
something else?

I think you meant the next "interesting" bit, but I don't know what
your definition of "interesting" is.

> move the pointer, and
> interrupt if the stream becomes empty. This is a TOOL; for some
> purposes, it will be used more frequently than additions.

But I live outside North America. I'm not allowed to use such tools.
[At least I _think_ you are alluding to cryptography.]

> To illustrate what one can do with it, suppose we want to generate
> random variables with density 4*x*(1-x) on (0,1). Generate two
> random variables A and B as above.

You mean using the code you posted, or some equivalent?

> Let K be A/2, rounded up, and

Rounded up to an integer? Since the extrema of your distribution were
0 and 1, this would make the integer K almost always 1, and very
occasionally 0 (when A = 0).

> let N be K+B.

Is N also to be an integer? If not, what does "N-th bit to the right"
mean? If so, since B is potentially fractional is N rounded, forced up
or truncated to an integer value?

> Then change the N-th bit to the right of the binary
> point of a uniform (0,1) random variable X to the opposite of the
> K-th to get the result. BTW, I believe you can see that to do this
> with a floating point representation of X is quite difficult.

Yes, it means denormalizing the floating point number, possibly
causing underflow if X is very small.

Do you want all of this implemented too? The little-endian byte
arrangement of the Intel could make it rather ugly, but it is
possible.

Regards

Dave

Subject: Re: Random numbers for C: Improvements.
Date: 19 Jan 1999 20:30:00 -0500
From: hrubin@odds.stat.purdue.edu (Herman Rubin)
Message-ID: <783bio$12vk@odds.stat.purdue.edu>
References: <cUwZzZKf1ety-pn2-deRYWonQPOXL@localhost>
Newsgroups: sci.stat.math,sci.math,sci.math.num-analysis
Lines: 117

In article <cUwZzZKf1ety-pn2-deRYWonQPOXL@localhost>,
David W. Noon <dwnoon@compuserve.com> wrote:
>On Tue, 19 Jan 1999 00:27:13, hrubin@odds.stat.purdue.edu (Herman
>Rubin) wrote:

>> >In article <cUwZzZKf1ety-pn2-vB2YCWNn5fgR@mfs-pci-bqh-vty163.as.wcom.net>,
>> >David W. Noon <dwnoon@compuserve.com> wrote:

 [snip]

>> >)This is very admirable, but it is basically saying that the code
>> >)should be in assembler. There goes portability.

>> Decent random number code cannot be portable. Anything which involves
>> bit handling, as this should, suffers from the problem. And besides
>> the problem of portability of code, there is the portability of results.
>> The order of generating two non-uniform random numbers, or the change
>> of size of a buffer, can do this with ease.

>I meant portable across software platforms, not hardware.

>I would hope your code executes on any 32-bit Intel system the same
>way, whether that system is running Windows NT, OS/2, Linux, any UNIX
>variant or whatever.

It is by no means clear that HLL generated code will do this, as the
machine instructions involved are different. I cannot see that there
is any more problem with the current assemblers, or with more
intelligently designed ones. The design changes should be for
efficiency on the part of the programmer, not the machine.

>> >)I don't mind translating all this stuff into assembler, but anybody
>> >)not using an Intel 80486 or better running a 32-bit version of OS/2
>> >)will not be able to use my code.

>> >)The only other language I know of that has sufficiently powerful macro
>> >)facilities to force inlining like C but with better syntax, is
>> >)portable across many platforms, and produces efficient object code is
>> >)PL/I. If anybody besides me can use such code, I will produce it

>> I doubt that PL/I can do the job. But I would welcome trying it.

>It can certainly handle the code that you have posted so far, and with
>ease.

>> Let me give you an important tool; this is to generate the distance
>> to the next bit in a stream of random bits,

What was meant was the distance to the next one in a stream of
random bits. In other words, generate a geometric (.5) random
variable using only the number of bits required by information.

>Surely the distance to the next bit is always 1 position. Did you mean
>something else?

>I think you meant the next "interesting" bit, but I don't know what
>your definition of "interesting" is.

>> move the pointer, and
>> interrupt if the stream becomes empty. This is a TOOL; for some
>> purposes, it will be used more frequently than additions.

>But I live outside North America. I'm not allowed to use such tools.
>[At least I _think_ you are alluding to cryptography.]

What does this have to do with cryptography? There are many uses for
this instruction in the computer literature, although none of the others
I have seen would use it that much. It is used for locating the next
record, where the positions of records are stored in single bits.

>> To illustrate what one can do with it, suppose we want to generate
>> random variables with density 4*x*(1-x) on (0,1). Generate two
>> random variables A and B as above.

>You mean using the code you posted, or some equivalent?

>> Let K be A/2, rounded up, and

>Rounded up to an integer? Since the extrema of your distribution were
>0 and 1, this would make the integer K almost always 1, and very
>occasionally 0 (when A = 0).

With the correction, A and B are positive integers; there is no
real problem.

>> let N be K+B.

>Is N also to be an integer? If not, what does "N-th bit to the right"
>mean? If so, since B is potentially fractional is N rounded, forced up
>or truncated to an integer value?

>> Then change the N-th bit to the right of the binary
>> point of a uniform (0,1) random variable X to the opposite of the
>> K-th to get the result. BTW, I believe you can see that to do this
>> with a floating point representation of X is quite difficult.

>Yes, it means denormalizing the floating point number, possibly
>causing underflow if X is very small.

In fact, it cannot, unless the mantissa of the floating point number
is exceptionally long. On most machines, it will have to be done
in the integer registers, anyhow. And they may not be long enough;
but one does not always need that accurate random numbers. At worst,
a bit past the end of the register will be targeted to be 1, and the
mantissa bits all cleared.

>Do you want all of this implemented too? The little-endian byte
>arrangement of the Intel could make it rather ugly, but it is
>possible.

It will not be as bad as that, but infinite precision methods are
not well suited to short registers.
--
This address is for information only. I do not claim that these views
are those of the Statistics Department or of Purdue University.
Herman Rubin, Dept. of Statistics, Purdue Univ., West Lafayette IN47907-1399
hrubin@stat.purdue.edu Phone: (765)494-6054 FAX: (765)494-0558

Subject: Re: Random numbers for C: Improvements.
Date: Tue, 19 Jan 1999 18:09:38 -0500
From: dwnoon@compuserve.com (David W. Noon)
Message-ID: <cUwZzZKf1ety-pn2-FJ7g8dchLbd1@localhost>
References: <780843ikj1@relay1.dsccc.com>
Newsgroups: sci.stat.math,sci.math,sci.math.num-analysis
Lines: 48

On Mon, 18 Jan 1999 21:12:35, jmccarty@sun1307.spd.dsccc.com (Mike
McCarty) wrote:

> In article <cUwZzZKf1ety-pn2-vB2YCWNn5fgR@mfs-pci-bqh-vty163.as.wcom.net>,
> David W. Noon <dwnoon@compuserve.com> wrote:
>
> [snip]
>
>)This is very admirable, but it is basically saying that the code
>)should be in assembler. There goes portability.
>)
>)I don't mind translating all this stuff into assembler, but anybody
>)not using an Intel 80486 or better running a 32-bit version of OS/2
>)will not be able to use my code.
>)
>)The only other language I know of that has sufficiently powerful macro
>)facilities to force inlining like C but with better syntax, is
>)portable across many platforms, and produces efficient object code is
>)PL/I. If anybody besides me can use such code, I will produce it
>
> You haven't heard of C++?

Of course I have. I earn my living writing C++, including the
reprehensible MFC when the client demands it.

C++ just isn't ideally suited to the task under current
implementations of the language. I have already suggested (in another
post in this thread) templates as a substitute for method calls, to
try and get some speed improvement, but generally C++ compilers
produce code that is too bloated and too slow for high-volume number
crunching. [You might note the expression "produces efficient object
code" in my original post, which you have quoted above.]

You might ask yourself why "low level" coders use their C/C++
compilers in C mode. It is definitely not for more elegant syntax!

FORTRAN doesn't have a macro facility in present implementations.
Neither does Pascal or ALGOL 68. I have compilers for all of these,
but they won't do slick inlining.

So, what does that leave? COBOL? No, it lacks macros too.

That leaves the systems programmer's two faithful friends: PL/I and
assembler.

Regards

Dave

Subject: Re: Random numbers for C: Improvements.
Date: 20 Jan 1999 07:49:11 GMT
From: zaykin@statgen.ncsu.edu (Dmitri Zaykin)
Message-ID: <7841pn$1ha$1@uni00nw.unity.ncsu.edu>
References: <cUwZzZKf1ety-pn2-FJ7g8dchLbd1@localhost>
Newsgroups: sci.stat.math,sci.math,sci.math.num-analysis
Lines: 15

In sci.stat.math David W. Noon <dwnoon@compuserve.com> wrote:
> I have already suggested (in another post in this thread) templates
> as a substitute for method calls, to try and get some speed
> improvement (...)

There are no calls for inlined methods.

Templates can be an alternative to C-macros in terms of
genericity, not inlining. C++ templates do not have to be
inlined. They do not offer anything in terms of speed
except as a substitute for C++ inheritance (again, as a
faster tool for doing generic stuff) which is not an
issue here.

Dmitri

Subject: Re: Random numbers for C: Improvements.
Date: 18 Jan 1999 21:11:32 GMT
From: jmccarty@sun1307.spd.dsccc.com (Mike McCarty)
Message-ID: <780824$ie4$1@relay1.dsccc.com>
References: <77t9ot$15ha@b.stat.purdue.edu>
Newsgroups: sci.math,sci.math.num-analysis,sci.stat.math
Lines: 51

In article <77t9ot$15ha@b.stat.purdue.edu>,
Herman Rubin <hrubin@b.stat.purdue.edu> wrote:

[snip]

)I think it should be clarified, and probably written out in some
)more detail. But the procedure call overhead would be comparable
)to the computing cost; C, and all other languages, have such great
)built-in inefficiencies that what is needed is something written
)from the standpoint of mathematics and efficiency.

What do you mean by this? *No* language has built in inefficiencies. I
defy you to find anywhere in the ANSI/ISO C definition *any* statement
about something being no more than so-and-so efficient.

Is assembler a language?

The efficiency of emitted code is a matter of quality of
implementation, not of language. I wrote a compiler several years ago,
intended for sale (but the project was cancelled) which automatically
inlined procedures (or functions) which were called only once. The
first target language for sale (it was to have different front ends)
was C. C++ has a specific means for requesting that a function be
expanded inline in every invocation.

Now some *processors* are more efficient than others, but I don't think
you mean that.

)But even the use of a comparison is a costly operation, if the
)result is to be used promptly. Any form of random, pseudo-random,
)or quasi-random numbers should be done in good integer form,

An amazing statement, given that there are machines available today
which do floating arithmetic faster than integer.

)not present now on many machines because of bad design coming
)from bad languages which have thrown out the natural properties
)of computers, and attempts to keep stupid people from making
)mistakes. An idiot-proof language is only for idiots.

This last statement makes no sense to me at all. There are no
"idiot-proof languages". Some languages have more checking built into
them, but this in no way should affect the quality of the generated
code.

Mike
--

```
----
char *p="char *p=%c%s%c;main(){printf(p,34,p,34);}";main(){printf(p,34,p,34);}
This message made from 100% recycled bits.
I don't speak for Alcatel      &LT;- They make me say that.
```

Subject: Re: Random numbers for C: Improvements.
Date: 18 Jan 1999 19:14:30 -0500
From: hrubin@odds.stat.purdue.edu (Herman Rubin)
Message-ID: <780ip6$11h8@odds.stat.purdue.edu>
References: <780824$ie4$1@relay1.dsccc.com>
Newsgroups: sci.math,sci.math.num-analysis,sci.stat.math
Lines: 88

In article <780824$ie4$1@relay1.dsccc.com>,
Mike McCarty <jmccarty@sun1307.spd.dsccc.com> wrote:
>In article <77t9ot$15ha@b.stat.purdue.edu>,
>Herman Rubin <hrubin@b.stat.purdue.edu> wrote:

[snip]

>)I think it should be clarified, and probably written out in some
>)more detail. But the procedure call overhead would be comparable
>)to the computing cost; C, and all other languages, have such great
>)built-in inefficiencies that what is needed is something written
>)from the standpoint of mathematics and efficiency.

>What do you mean by this? *No* language has built in inefficiencies. I
>defy you to find anywhere in the ANSI/ISO C definition *any* statement
>about something being no more than so-and-so efficient.

A language has a built in inefficiency in every situation in which
the optimal use of machine instructions cannot be addressed in that
language. It has inefficiencies in all situations where workarounds
need to be made.

>Is assembler a language?

Currently, a very difficult one to use. This is because the computer
people have deliberately tried to make it so. There is no reason why
a user cannot reconfigure the "mnemonics" (they are generally atrocious)
so that they can be easily written.

>The efficiency of emitted code is a matter of quality of
>implementation, not of language. I wrote a compiler several years ago,
>intended for sale (but the project was cancelled) which automatically
>inlined procedures (or functions) which were called only once. The
>first target language for sale (it was to have different front ends)
>was C. C++ has a specific means for requesting that a function be
>expanded inline in every invocation.

More than this is needed. The implementations do not take into
account what the intelligent programmer knows; they will not allow
this input.

>Now some *processors* are more efficient than others, but I don't think
>you mean that.

No, I do not.

>)But even the use of a comparison is a costly operation, if the
>)result is to be used promptly. Any form of random, pseudo-random,
>)or quasi-random numbers should be done in good integer form,

>An amazing statement, given that there are machines available today
>which do floating arithmetic faster than integer.

This is a design catastrophe, partly due to the bad languages. There
is no such thing as floating point architecture, but architecture
which does a combination of operations for badly designed floating
point arithmetic. With almost no more cost, this arithmetic could
be made available for fixed point. BTW, there is great need for
fixed point non-integer arithmetic. And if increased precision is
needed, it is necessary to emulate fixed point with floating, which
is quite clumsy.

In addition, the fundamental random input should be a bit stream.
Try using such. The hardware to do it in reasonable time is not
present on any machine.

>)not present now on many machines because of bad design coming
>)from bad languages which have thrown out the natural properties
>)of computers, and attempts to keep stupid people from making
>)mistakes. An idiot-proof language is only for idiots.

>This last statement makes no sense to me at all. There are no
>"idiot-proof languages". Some languages have more checking built into
>them, but this in no way should affect the quality of the generated
>code.

I suggest that this checking be made strictly voluntary. I know
what I want the computer to do, and how to use machine instructions
to do it, IF I can get a list of the machine instructions. The
language has no such capability.

Checking is extremely expensive. It is now largely a matter of
hardware; user hardware-style interrupts are needed.
--
This address is for information only. I do not claim that these views
are those of the Statistics Department or of Purdue University.
Herman Rubin, Dept. of Statistics, Purdue Univ., West Lafayette IN47907-1399
hrubin@stat.purdue.edu Phone: (765)494-6054 FAX: (765)494-0558

Subject: Re: Random numbers for C: Improvements.
Date: 21 Jan 1999 19:07:19 GMT
From: jmccarty@sun1307.spd.dsccc.com (Mike McCarty)
Message-ID: <787tt7gr1@relay1.dsccc.com>
References: <780ip6$11h8@odds.stat.purdue.edu>
Newsgroups: sci.math,sci.math.num-analysis,sci.stat.math
Lines: 34

In article <780ip6$11h8@odds.stat.purdue.edu>,
Herman Rubin <hrubin@odds.stat.purdue.edu> wrote:
)In article <780824$ie4$1@relay1.dsccc.com>,
)Mike McCarty <jmccarty@sun1307.spd.dsccc.com> wrote:
)>In article <77t9ot$15ha@b.stat.purdue.edu>,
)>Herman Rubin <hrubin@b.stat.purdue.edu> wrote:

[snip]

Much of what you wrote here struck me as unsubstantiated opinion. None
of it appeared to be worth attempting to argue over.

)>This last statement makes no sense to me at all. There are no
)>"idiot-proof languages". Some languages have more checking built into
)>them, but this in no way should affect the quality of the generated
)>code.
)
)I suggest that this checking be made strictly voluntary. I know
)what I want the computer to do, and how to use machine instructions
)to do it, IF I can get a list of the machine instructions. The
)language has no such capability.
)
)Checking is extremely expensive. It is now largely a matter of
)hardware; user hardware-style interrupts are needed.

The checking I had in mind costs *nothing* during the execution of the
program.

Mike
--
---
char *p="char *p=%c%s%c;main(){printf(p,34,p,34);}";main(){printf(p,34,p,34);}
This message made from 100% recycled bits.
I don't speak for Alcatel <- They make me say that.

Subject: Re: Random numbers for C: Improvements.
Date: 16 Jan 1999 17:56:05 -0500
From: zaykin@statgen.ncsu.edu (Dmitri Zaykin)
Message-ID: <7k8ym6d4q.fsf@poole.statgen.ncsu.edu>
References: <369F6FCA.74C7C041@stat.fsu.edu>
Newsgroups: sci.stat.math,sci.math,sci.math.num-analysis
Lines: 12

I see one more problem with the code.

The table is declared as "t[256]". Then expressions like t[++c] are
safe assuming that the unsigned char c "wraps around" and becomes zero
after 255.

However, expressions like c+237 evaluate to integer and have to be
casted to the unsigned char explicitly, e.g. "t[(unsigned char)(c+237)]",
should be used instead of t[c+237] (the later is accessing beyond the
array bounds for c>18).

Dmitri

Subject: Re: Random numbers for C (and assembly)
Date: Tue, 19 Jan 1999 11:10:17 GMT
From: qscgz@my-dejanews.com
Message-ID: <781p6llrv1@nnrp1.dejanews.com>
References: <369F6FCA.74C7C041@stat.fsu.edu>
Newsgroups: sci.stat.math,sci.math,sci.math.numer-analysis
Lines: 58

George Marsaglia wrote about these PRNGs:

>MWC:((z=36969*(z&65535)+(z>>16))<<16)+((w=18000*(w&65535)+(w>>16))&65535)
>SHR3:(jsr=(jsr=(jsr=jsr^(jsr<<17))^(jsr>>13))^(jsr<<5))
>CONG:(jcong=69069*jcong+1234567)
>LFIB4:(t[c]=t[c]+t[c+58]+t[c+119]+t[c+178],t[++c])
>SWB:(t[c+237]=(x=t[c+15])-(y=t[c]+(x<y)),t[++c])

    DIEHARD-tests fail on :

```
MWC (z only) :  none
SHR3 : brank,opso,oqso,dna,count
LFIB4: none
CONG (16 highbits) : opso,oqso,dna
SWB: bday,brank,count


In x86-assembly :( using:  a for eax , c for ecx  ,  _  for sbb , # for adc )

  bytes required  |  MWC(8):ax=36969  ax*mwc[4]  ax+mwc[2]  dx#0  mwc[4]=ax
mwc[2]=dx  ax=18000  ax*[mwc]  ax+mwc[6]  dx#0  [mwc]=ax  mwc[6]=dx
SHR3(4):a=[shr3]  c=a  a&LT; &LT; 17    a^c  c=a  a>>13  a^c  c=a  a&LT; &LT; 5  a^c
[shr3]=a  CONG(4):a=69069  a*[cong]  a+1234567  [cong]=a
LFIB4(1032):c=lfib4[4]  a=lfib4[4c+8]  cl+58  a+lfib4[4c+8]  cl+61
a+lfib4[4c+8]  cl+60  a+lfib4[4c+8]  cl+77  lfib4[4c+8]=a  cl+1  lfib4[4]=c
[lfib4]=a  SWB(1032):c=swb[4]  cl+15  a=swb[4c+8]  cl-14  a_swb[4c+8]  cl+236
 swb[4c+8]=a  cl-236  swb[4]=c  [swb]=a


I estimate:
    29,8,12,12,10  cyles  on a P5   for  MWC,SHR3,CONG,LFIB4,SWB
    13,7,7,8,6     cycles on a P6 (P-II)

optimized with some calculations in parallel:
    23,6,11,8,5    cycles on a P5
    7,4,3,7,4      cycles on a P6

(these are only untested estimates)

for compound generators add the cycles of the parts.

( on a P6, the LFIB4 and SWB code should be changed (no cl),
     to avoid partial register stalls (slow). )

what is the fastest PRNG , that passes all tests ?
I think that 1 or 2 bytes per cycle are possible.
So, you could call a small assembly routine to fill an
array with ~500 random-bytes in 1microsec. and then use
it in C, if you prefer C.



qscgz@aol.com
```

```
Subject: Random numbers for C: The END?
Date: Wed, 20 Jan 1999 10:55:14 -0500
From: George Marsaglia &LT;geo@stat.fsu.edu>
Message-ID: &LT;36A5FC62.17C9CC33@stat.fsu.edu>
Newsgroups: sci.stat.math,sci.math
Lines: 301

My offer of RNG's for C was an invitation to dance;
I did not expect the Tarantella.  I hope this post will
stop the music, or at least slow it to a stately dance
for language chauvinists and software police---under
a different heading.

In response to a number of requests for good RNG's in
C, and mindful of the desirability of having a variety
of methods readily available, I offered several. They
were implemented as in-line functions using the #define
feature of C.

Numerous responses have led to improvements; the result
is the listing below, with comments describing the
generators.

I thank all the experts who contributed suggestions, either
directly to me or as part of the numerous threads.

It seems necessary to use a (circular) table in order
to get extremely long periods for some RNG's. Each new
number is some combination of the previous r numbers, kept
in the circular table.  The circular table has to keep
at least the last r, but possible more than r, numbers.

For speed, an  8-bit index seems best for accessing
members of the table---at least for Fortran, where an
8-bit integer is readily  available via integer*1, and
arithmetic on the index is automatically mod 256
(least-absolute-residue).

Having little experience with C, I got out my little
(but BIG) Kernighan and Ritchie book to see if there
were an 8-bit integer type. I found none, but I did
find char and unsigned char: one byte. Furthemore, K&R
said arithmetic on characters was ok. That, and a study
of the #define examples, led me to propose #define's
for in-line generators LFIB4 and SWB, with monster
periods. But it turned out that char arithmetic jumps
"out of character", other than for simple cases such as
c++ or c+=1.   So, for safety, the index arithmetic
below is kept in character by the UC definition.
```

Another improvement on the original version takes advantage of the comma operator, which, to my chagrin, I had not seen in K&R. It is there, but only with an example of (expression,expression). From the advice of contributors, I found that the comma operator allows (expression,...,expression,expression) with the last expression determining the value. That makes it much easier to create in-line functions via #define (see SHR3, LFIB4, SWB and FIB below).

The improved #define's are listed below, with a function to initialize the table and a main program that calls each of the in-line functions one million times and then compares the result to what I got with a DOS version of gcc. That main program can serve as a test to see if your system produces the same results as mine.

```
 _____________________________________
|If you run the program below, your output|
| should be  seven lines, each a 0 (zero).|
 ----------------------------------------
```

Some readers of the threads are not much interested in the philosophical aspects of computer languages, but want to know: what is the use of this stuff? Here are simple examples of the use of the in-line functions: Include the #define's in your program, with the accompanying static variable declarations, and a procedure, such as the example, for initializing the static variable (seeds) and the table.

Then any one of those in-line functions, inserted in a C expression, will provide a random 32-bit integer, or a random float if UNI or VNI is used. For example, KISS&255; would provide a random byte, while 5.+2.*UNI; would provide a random real (float) from 5 to 7. Or 1+MWC%10; would provide the proverbial "take a number from 1 to 10", (but with not quite, but virtually, equal probabilities).
More generally, something such as 1+KISS%n; would provide a practical uniform random choice from 1 to n, if n is not too big.

A key point is: a wide variety of very fast, high-quality, easy-to-use RNG's are available by means of the nine in-line functions below, used individually or in combination.

The comments after the main test program describe the generators. These descriptions are much as in the first post, for those who missed them. Some of the generators (KISS, MWC, LFIB4) seem to pass all tests of randomness, particularly the DIEHARD battery of tests, and combining virtually any two or more of them should provide fast, reliable, long period generators. (CONG or FIB alone and CONG+FIB are suspect, but quite useful in combinations.)

Serious users of random numbers may want to run their simulations with several different generators, to see if they get consistent results. These #define's may make it easy to do.
Bonne chance,
George Marsaglia

The C code follows----------:

```
#include &LT;stdio.h>
#define znew   (z=36969*(z&65535)+(z>>16))
#define wnew   (w=18000*(w&65535)+(w>>16))
#define MWC    ((znew&LT;&LT;16)+wnew )
#define SHR3  (jsr^=(jsr&LT;&LT;17), jsr^=(jsr>>13), jsr^=(jsr&LT;&LT;5))
#define CONG  (jcong=69069*jcong+1234567)
#define FIB   ((b=a+b),(a=b-a))
#define KISS  ((MWC^CONG)+SHR3)
#define LFIB4 (c++,t[c]=t[c]+t[UC(c+58)]+t[UC(c+119)]+t[UC(c+178)])
#define SWB   (c++,bro=(x&LT;y),t[c]=(x=t[UC(c+34)])-(y=t[UC(c+19)]+bro))
#define UNI   (KISS*2.328306e-10)
#define VNI   ((long) KISS)*4.656613e-10
#define UC    (unsigned char)  /*a cast operation*/
typedef unsigned long UL;

/*  Global static variables: */
 static UL z=362436069, w=521288629, jsr=123456789, jcong=380116160;
 static UL a=224466889, b=7584631, t[256];
/* Use random seeds to reset z,w,jsr,jcong,a,b, and the table t[256]*/

 static UL x=0,y=0,bro; static unsigned char c=0;

/* Example procedure to set the table, using KISS: */
 void settable(UL i1,UL i2,UL i3,UL i4,UL i5, UL i6)
 { int i; z=i1;w=i2,jsr=i3; jcong=i4; a=i5; b=i6;
 for(i=0;i&LT;256;i=i+1)  t[i]=KISS;
 }
```

```
/* This is a test main program.  It should compile and print 7  0's. */
int main(void){
int i; UL k;
settable(12345,65435,34221,12345,9983651,95746118);

for(i=1;i&LT;1000001;i++){k=LFIB4;} printf("%u\n", k-1064612766U);
for(i=1;i&LT;1000001;i++){k=SWB  ;} printf("%u\n", k- 627749721U);
for(i=1;i&LT;1000001;i++){k=KISS ;} printf("%u\n", k-1372460312U);
for(i=1;i&LT;1000001;i++){k=CONG ;} printf("%u\n", k-1529210297U);
for(i=1;i&LT;1000001;i++){k=SHR3 ;} printf("%u\n", k-2642725982U);
for(i=1;i&LT;1000001;i++){k=MWC  ;} printf("%u\n", k- 904977562U);
for(i=1;i&LT;1000001;i++){k=FIB  ;} printf("%u\n", k-3519793928U);
             }
/*-------------------------------------------------------
   Write your own calling program and try one or more of
   the above, singly or in combination, when you run a
   simulation. You may want to change the simple 1-letter
   names, to avoid conflict with your own choices.        */

/* All that follows is comment, mostly from the initial
   post. You may want to remove it */

/* Any one of KISS, MWC, FIB, LFIB4, SWB, SHR3, or CONG
   can be used in an expression to provide a random 32-bit
   integer.

   The KISS generator, (Keep It Simple Stupid), is
   designed to combine the two multiply-with-carry
   generators in MWC with the 3-shift register SHR3 and
   the congruential generator CONG, using addition and
   exclusive-or. Period about 2^123.
   It is one of my favorite generators.

   The  MWC generator concatenates two 16-bit multiply-
   with-carry generators, x(n)=36969x(n-1)+carry,
   y(n)=18000y(n-1)+carry  mod 2^16, has period about
   2^60 and seems to pass all tests of randomness. A
   favorite stand-alone generator---faster than KISS,
   which contains it.

   FIB is the classical Fibonacci sequence
   x(n)=x(n-1)+x(n-2),but taken modulo 2^32.
   Its period is 3*2^31 if one of its two seeds is odd
   and not 1 mod 8. It has little worth as a RNG by
   itself, but provides a simple and fast component for
   use in combination generators.

   SHR3 is a 3-shift-register generator with period
   2^32-1. It uses y(n)=y(n-1)(I+L^17)(I+R^13)(I+L^5),
   with the y's viewed as binary vectors, L the 32x32
   binary matrix that shifts a vector left 1, and R its
   transpose.  SHR3 seems to pass all except those
   related to the binary rank test, since 32 successive
   values, as binary vectors, must be linearly
   independent, while 32 successive truly random 32-bit
   integers, viewed as binary vectors, will be linearly
   independent only about 29% of the time.

   CONG is a congruential generator with the widely used 69069
   multiplier: x(n)=69069x(n-1)+1234567.  It has period
   2^32. The leading half of its 32 bits seem to pass
   tests, but bits in the last half are too regular.

   LFIB4 is an extension of what I have previously
   defined as a lagged Fibonacci generator:
   x(n)=x(n-r) op x(n-s), with the x's in a finite
   set over which there is a binary operation op, such
   as +,- on integers mod 2^32, * on odd such integers,
   exclusive-or(xor) on binary vectors. Except for
   those using multiplication, lagged Fibonacci
   generators fail various tests of randomness, unless
   the lags are very long. (See SWB below).
   To see if more than two lags would serve to overcome
   the problems of 2-lag generators using +,- or xor, I
   have developed the 4-lag generator LFIB4 using
   addition: x(n)=x(n-256)+x(n-179)+x(n-119)+x(n-55)
   mod 2^32. Its period is 2^31*(2^256-1), about 2^287,
   and it seems to pass all tests---in particular,
   those of the kind for which 2-lag generators using
   +,-,xor seem to fail.  For even more confidence in
   its suitability,  LFIB4 can be combined with KISS,
   with a resulting period of about 2^410: just use
   (KISS+LFIB4) in any C expression.

   SWB is a subtract-with-borrow generator that I
   developed to give a simple method for producing
   extremely long periods:
      x(n)=x(n-222)-x(n-237)- borrow mod 2^32.
   The 'borrow' is 0, or set to 1 if computing x(n-1)
   caused overflow in 32-bit integer arithmetic. This
   generator has a very long period, 2^7098(2^480-1),
   about 2^7578.   It seems to pass all tests of
   randomness, except for the Birthday Spacings test,
   which it fails badly, as do all lagged Fibonacci
```

```
generators using +,- or xor. I would suggest
combining SWB with KISS, MWC, SHR3, or CONG.
KISS+SWB has period >2^7700 and is highly
recommended.
Subtract-with-borrow has the same local behaviour
as lagged Fibonacci using +,-,xor---the borrow
merely provides a much longer period.
SWB fails the birthday spacings test, as do all
lagged Fibonacci and other generators that merely
combine two previous values by means of =,- or xor.
Those failures are for a particular case: m=512
birthdays in a year of n=2^24 days. There are
choices of m and n for which lags >1000 will also
fail the test.  A reasonable precaution is to always
combine a 2-lag Fibonacci or SWB generator with
another kind of generator, unless the generator uses
*, for which a very satisfactory sequence of odd
32-bit integers results.

The classical Fibonacci sequence mod 2^32 from FIB
fails several tests.  It is not suitable for use by
itself, but is quite suitable for combining with
other generators.

The last half of the bits of CONG are too regular,
and it fails tests for which those bits play a
significant role. CONG+FIB will also have too much
regularity in trailing bits, as each does. But keep
in mind that it is a rare application for which
the trailing bits play a significant role.  CONG
is one of the most widely used generators of the
last 30 years, as it was the system generator for
VAX and was incorporated in several popular
software packages, all seemingly without complaint.

Finally, because many simulations call for uniform
random variables in 0&LT;x&LT;1 or -1&LT;x&LT;1, I use #define
statements that permit inclusion of such variates
directly in expressions:  using UNI will provide a
uniform random real (float) in (0,1), while VNI will
provide one in (-1,1).

All of these: MWC, SHR3, CONG, KISS, LFIB4, SWB, FIB
UNI and VNI, permit direct insertion of the desired
random quantity into an expression, avoiding the
time and space costs of a function call. I call
these in-line-define functions.  To use them, static
variables z,w,jsr,jcong,a and b should be assigned
seed values other than their initial values.  If
LFIB4 or SWB are used, the static table t[256] must
be initialized.

A note on timing:  It is difficult to provide exact
time costs for inclusion of one of these in-line-
define functions in an expression.  Times may differ
widely for different compilers, as the C operations
may be deeply nested and tricky. I suggest these
rough comparisons, based on averaging ten runs of a
routine that is essentially a long loop:
for(i=1;i&LT;10000000;i++) L=KISS; then with KISS
replaced with SHR3, CONG,... or KISS+SWB, etc. The
times on my home PC, a Pentium 300MHz, in nanoseconds:
FIB 49;LFIB4 77;SWB 80;CONG 80;SHR3 84;MWC 93;KISS 157;
VNI 417;UNI 450;
*/
```

Terry Ritter, his current address, and his top page.

Last updated: 1999-02-18

The Hardware Random Number Generator

A *Ciphers By Ritter* Page

The conversation starts with a particular prescription for a physically-random generator. It then breaks into mostly theory, with a few comments on alternate approaches.

Contents

- 1999-01-22 .·´¯`·..·´¯`·..·´¯`·..·´¯`·..·´¯`·..·´¯`·.. "Those who seek to capture random numbers in their natural habitats often are faced with two difficult paths: to wait quietly for the small whispers of thermal noise or to become immersed in the dazzling thrashes of large signal sources."
- 1999-01-24 Terry Ritter: "Doing low-level analog in a digital environment is tough, there is no doubt about that. Presumably we would employ appropriate techniques to pull it off."
- 1999-01-24 .·´¯`·..·´¯`·..·´¯`·..·´¯`·..·´¯`·..·´¯`·.. "A digital ring oscillator composed of Schmitt Triggers (with hysteresis) can be designed to have a slow rise time of ten microseconds, but they respond to an input transition in 100ps, to use round numbers."
- 1999-01-24 Terry Ritter: "...this 'large signal' design is probably sensitive to even tiny power and ground transients." "What I see is a huge complexity-based increase in signal transitions (a frequency increase) which will be hard to distinguish from heat-based noise. And if we cannot distinguish operation *with* noise from operation *without* noise, we have no way to prove that noise is involved at all." "Even PSEUDO-random RNG's pass statistical tests. Those tests have nothing to do with cryptographic unpredictability or 'strength.'"
- 1999-01-24 R. Knauer: "That statement needs to be added to the FAQ on Crypto-Grade Randomness."
- 1999-01-02 jsavard@ecn.ab.ca: "Many people have heard this, but because they have not understood, they did not believe."
- 1999-01-25 R. Knauer: "Here is a post from Patrick Juola that expands on this in a way that can be understood by all."
- 1999-01-25 .·´¯`·..·´¯`·..·´¯`·..·´¯`·..·´¯`·..·´¯`·.. "The major source of randomness of this RNG is the unsynchronized nature of multiple oscillators with randomly changing frequencies. This is a large signal phenomenon, which cannot be accurately described mathematically."
- 1999-01-25 Trevor Jackson, III: "Since the (intermediate) output of the system is driving the changes to the oscillators there is a full feedback loop present. This kind of system may pass statistical tests for randomness, but it may not be unpredictable."
- 1999-01-26 handWave: "We ran mathematical auto-correlation tests looking exactly for this, and got good results."
- 1999-01-26 Mok-Kong Shen: "If I have two sources of randomness, one software and one hardware, both passing all statistical tests I apply equally well, why should I choose one source in preference to the other?"
- 1999-01-26 Trevor Jackson, III: "We cannot re-run the hardware and get the same output. Thus the hardware is superior" "Everyone alive in 1980 knew that software was the 'plastic' of the decade and that the market for software was going to grow quickly. But no other person alive in 1980 forsaw just how big the market would be for really bad software."
- 1999-01-26 R. Knauer: "...the auto industry knew that way before Gates used it in the S/W industry."
- 1999-01-27 Mok-Kong Shen: "If you produce some sequence with a sufficiently good algorithm with a sufficiently long key and later forget that key, even you wouldn't be able to reproduce the sequence."
- 1999-01-26 R. Knauer: "Why do you persist in believing that statistical tests have anything to do with randomness in cryptography?"
- 1999-01-26 Mok-Kong Shen: "Tell me what other (better) tools are available for me to make the decision."
- 1999-01-26 R. Knauer: "If I told you that there are none, would you believe me?" "The key to understanding is that randomness depends on the generation process, not the numbers themselves."
- 1999-01-27 Mok-Kong Shen: "Then tell me HOW to get such a physical device that PROVABLY is capable of generating all possible sequences of a given finite length equiprobalbly."
- 1999-01-27 Patrick Juola: "You can't. Tell me how you can build a plane that will *provably* fly equally stably in any direction."
- 1999-01-27 Mok-Kong Shen: "Since there can't be an good answer, one can't claim hardware sequences are always to be preferred to software sequences."
- 1999-01-27 Patrick Juola: "I can prove that there can't be a good s/w sequence running on a deterministic machine. But I can't do that merely by inspecting any finite sample of outputs -- I have to look at the generators to do it." "...one can't claim hardware sequences are always to be preferred to software sequences *on the basis of a statistical analysis of a finite set of output sequences.*"
- 1999-01-27 Mok-Kong Shen: "Are there other sound scientific basis to claim the said preference."
- 1999-01-27 Patrick Juola: "...yes, if your goal is to provide unbounded degrees of security for messages of unbounded length."
- 1999-01-27 Mok-Kong Shen: "Could you provide the requested sound scientific basis?"
- 1999-01-27 Patrick Juola: "If the key is 'as random as' the message, then Shannon's proof of secrecy goes through." "The difficulty is in *proving* that a given sequence of N bits is contains N bits of randomness...."
- 1999-01-27 R. Knauer: "You once said that such a system was vulnerable to a Bayesian attack. Have you changed your mind?"
- 1999-01-28 Patrick Juola: "No. The key point here is that the key is as large as -- larger than, in fact -- the plaintext."
- 1999-01-28 R. Knauer: "Please elaborate on the concepts of 'bounded', 'unbounded'...."
- 1999-01-29 Patrick Juola: "The difference between bounded and finite is simple -- with finite, plaintexts, I know that my articles will eventually end, but I don't know when beforehand. With a bounded plantext, I set myself a rule beforehand that I won't go over 30 lines, or 300, or 3 million, whatever *and stick to that rule.*"
- 1999-01-28 Mok-Kong Shen: "We must be realistic, since theoretical stuffs may not be realizable in the real world and since 'absulute' security is never required.... Incidentally, in another thread I also suggested distiling bit sequences out of natural language texts as raw materials."
- 1999-01-27 R. Knauer:
- 1999-01-27 R. Knauer:
- 1999-01-27 R. Knauer: "The answer that the poster wants to hear is: Because TRNGs are not Perfect, PRNGs are just as good."
- 1999-01-27 Mok-Kong Shen: "Where is the proof...?"
- 1999-01-27 R. Knauer: "The proof of the second part comes from an analysis of what makes a PRNG behave the way it does."
- 1999-01-27 Mok-Kong Shen: "Please note I don't claim PRNGs are good. I simply doubt that hardward generators are good because I have no tools to determine that they are good, except by using statistical tools."
- 1999-01-27 R. Knauer: "You must be skilled at designing a TRNG. Statistical tools are worthless."
- 1999-01-27 Mok-Kong Shen: "...how do you show through a test of skill that the resulting TRNG designed by that person has a certain 'crypto-grade'...?"
- 1999-01-27 Patrick Juola: "The same way you test the skill of the architect who designed your building."
- 1999-01-27 R. Knauer: "Get several skilled people to check the design and the actual TRNG."
- 1999-01-27 Patrick Juola: "...if the generation process is purely software running on a deterministic routine, then the randomness of the resulting pad is bounded by the randomness of the starting conditions."
- 1999-01-27 Mok-Kong Shen: "...that cetainly doesn't imply that any hardware generator can offer absolute security."
- 1999-01-27 R. Knauer: "The OTP cryptosystem is proveably secure."
- 1999-01-27 Mok-Kong Shen: "Simply saying that a hardware generator offers high security is NO convincing argument...."
- 1999-01-27 R. Knauer: "Fine - let him seek a meaningful demonstration then."
- 1999-01-28 Mok-Kong Shen: "After having built a hardware generator, do you test it or simply 'believe' that it is o.k.?" "'If two objects are indistinguishable, in what sense are they different?'"
- 1999-01-28 R. Knauer:
- 1999-01-27 R. Knauer: "With correct design one can make a TRNG random to within an arbitrarily high level of precision."

- 1999-01-27 Mok-Kong Shen: "You have to define that 'precision' in scientific terms, in particular establish a 'unit' and provide a precise method to measure that 'precision' in that unit. Before that, you have nothing."
- 1999-01-27 R. Knauer: "You do that by analyzing the design, and performing certain tests on the actual equipment to make sure it meets the design specifications."
- 1999-01-27 Mok-Kong Shen: "Now what are the specifications that ensures a crypto-grade TRNG?"
- 1999-01-27 R. Knauer: "Check out the Hotbits radioactive decay TRNG...."
- 1999-01-27 Patrick Juola: "Well, broadly speaking, you need a source of randomness...." "Then, I need to sample those events in a manner that preserves the randomness." "The resulting number streams can be certified as "random," subject to the same (standard) assumptions that any other engineering project uses."
- 1999-01-27 Trevor Jackson, III: "Now this is an issue worthy of intense thought and debate.... I believe this breaks into two subtopics, one fundamentally describing the unit of measure and the other describing the measurement methodology."
- 1999-01-27 Medical Electronics Lab: "The "unit" is expectation value. Precision isn't a good term, what you want is "uniformity". A TRNG should have expectation values for *all* imaginable tests, i.e. it should be uniformly random no matter how you look at it."
- 1999-01-27 Patrick Juola: "Mr. Knauer's criterion...: "For all (finite) lengths, all possible sequences of that length are outputted equiprobable."
- 1999-01-27 R. Knauer: "...I see no problem with using the sequence...."
- 1999-01-29 Kurt Wismer: "this is the definition i've used for years...."
- 1999-01-29 Patrick Juola: "...it also looks suspiciously like the result of an incompetent engineer *trying* to build a RNG."
- 1999-01-30 Kurt Wismer: "i don't see any foolproof method of verifying that the trng is indeed a trng...."
- 1999-01-31 R. Knauer: "I believe that a radioactive decay TRNG can be verified to within a known level of precision."
- 1999-02-01 Patrick Juola: "But this is yet another example of a statistical test."
- 1999-02-03 R. Knauer: "The tests, statistical or otherwise, would be of a diagnostic nature applied to the sybsystems of the TRNG."
- 1999-02-03 Trevor Jackson, III: "So a properly designed RNG is not permitted to fails in an unknown way? Is this design methodology documented anywhere? It sounds like it avoids Murphy the way perpetual motion machines avoid friction."
- 1999-01-27 Patrick Juola: "You are going to have to open the hood and look at the engine and see how it's put together."
- 1999-01-27 R. Knauer: "Assuming that you could use such a Bayesian Test to characterize randomness, how does that differ from statistical tests for randomness?"
- 1999-01-27 Mok-Kong Shen: "What is a 'probability survey'?"
- 1999-01-27 R. Knauer: "I used the term to describe the process whereby one surveyed a large number of ciphers using probability techniques...."
- 1999-01-27 Patrick Juola: "By the time you get up to the upper reaches of what we can do with PRNGs, the information leak over reasonably-sizes samples is pretty infinitesimal." "Bayes' Theorem lets you refine hypotheses about biases that you've already made. Conventional statistics just let you test for the presence or absence of a visible bias."
- 1999-01-27 R. Knauer: "So, the Bayesian attack is not all that powerful against stream ciphers *in general*. You have to provide the first hypothesis to get it started."
- 1999-01-27 Trevor Jackson, III: "One can change keys arbitrarily. Once cannot change code so often...."
- 1999-01-27 R. Knauer: "How about making your algorithm (code) part of the key?"
- 1999-01-27 Patrick Juola: "The larger the key, the more difficulty you have in changing/exchanging it."
- 1999-01-27 R. Knauer: "The question was theoretical."
- 1999-01-27 Trevor Jackson, III: "...you need a clever encoding scheme such that all key values translate to valid algoirhms. In addition, you need to show that all such encoded algorithms are 'secure'."
- 1999-01-27 R. Knauer: "A daunting task."
- 1999-01-27 Patrick Juola: "...the NSA does *NOT* have infinite resources. Far from it."
- 1999-01-27 Trevor Jackson, III: "...analysis of the generation process is more efficient than analysis of the generator output."
- 1999-01-27 R. Knauer: "Algorithmic analysis of the generated output itself is completely worthless if that output is finite in length."
- 1999-01-27 Trevor Jackson, III: "Analysis of output is insufficient to prove the entropy density is 100%, but it can easily provide useful information."
- 1999-01-27 Patrick Juola: "...you *NEED* to define the tests you're going to run and the rejection threshhold before you power up the generator."
- 1999-01-28 Trevor Jackson, III: "Absolutely. Any other arrangement produces a man-in-the-loop situation."
- 1999-01-27 R. Knauer:
- 1999-01-28 Patrick Juola: "In terms of finite length numbers, a finite number is not statistically random if I can predict it (or something about it)."
- 1999-01-28 R. Knauer: "You throw out the suspect bad generators and argue that such a practice is safe. But what is your argument for the generators that you do not throw out?"
- 1999-01-29 Patrick Juola: "That the amount of the bias *that I measured* is less than some threshhold. If I can live with that threshhold and I believe that no other (untested) source of bias is likely to be present, then the cypher is safe to use."
- 1999-01-30 R. Knauer: "Then you accept bad generators and reject good ones."
- 1999-02-01 Patrick Juola: "Absolutely. With a particular, known, measured probability."
- 1999-02-03 R. Knauer: "...your tests will eventually reject every TRNG you test long enough...."
- 1999-02-03 Patrick Juola: "I 'know' that a given TRNG will fail 1/20 of the tests at a 5% level, so if I get an approximate failure rate of 1 in 20, then I keep that one...." "Statistics -- nor engineering practice, for that matter -- will never allow you to get a probability all the way down to zero."
- 1999-02-05 R. Knauer: "Just how much testing do you need to do to convince yourself that a given RNG is crypto-grade (either proveably secure or secure in terms of work effort)?"
- 1999-02-04 Herman Rubin: "If one can assume independence between generated sets, one can make both of these errors quite small...."
- 1999-02-04 R. Knauer: "I am interested in seeing how much testing is involved for a given level of confidence, and how that testing effort increases with decreasing levels of confidence."
- 1999-02-05 Patrick Juola: "Well, one starts out by making a questionable assumption, in most cases...."
- 1999-02-07 Herman Rubin: "...the sample size needed would increase as 1/d^2...." "Now if we XOR independent files, say run on different days, the deviations from randomness multiply."
- 1999-02-07 R. Knauer: "If your tests are probabilistic with only a 'good chance' of being correct, then how can they be relied on?"
- 1999-02-08 Patrick Juola: "What is your acceptable degree of uncertainty? "
- 1999-02-09 R. Knauer: "The only way to get a "minimal good chance" of characterizing the TRNG itself is to do a complete audit on it."
- 1999-02-09 Trevor Jackson, III: "...any judgement or decision is going to be based on some evidence gathered by observation and/or experiment." "Have you ever audited a system for side effects? Like exhaustive logic tests or provable software design, this kind of analysis is only useful for toy systems."
- 1999-02-08 Herman Rubin: "...the type of TRNG you think you have does not exist, only approximations of it can exist. And it is necessary to test if the approximations are good enough."
- 1999-02-08 R. Knauer: "...the very thing you are testing the RNG for, namely its suitability for use with the OTP system, is not determinable."
- 1999-02-08 Patrick Juola: "There are two things you need to do to produce a certifiable TRNG." "One is to confirm that the device is, in fact, a 'random number generator' in the sense that it produces random bits." "The other is to confirm that the outputs are bias-free -- or more accuratley as bias-free as possible...."
- 1999-02-09 R. Knauer: "We speak of the ills of bit-bias in terms of random number generation, but what if the generator were designed with a deliberate bias?"
- 1999-02-10 Trevor Jackson, III: "Any tests you would use to prove QM indeterminate can be used t prove a non QM RNG indeterminate. naturally, these would be statistical tests."
- 1999-02-10 R. Knauer:
- 1999-02-10 Patrick Juola: "If it's a really good generator with a known bias, there are mathematical techniques that will allow me to strip out the bias and produced an unbiased stream."
- 1999-02-11 Tony T. Warnock: "..."random" bits are quite expensive to get."
- 1999-02-11 R. Knauer: "This brings up the question of whether anti-skewing changes the equiprobability of the generator." "Anti-skewing procedures do not generate those sequences that under-representted because of the bias."

- 1999-02-11 Patrick Juola: "That's *exactly* what anti-skewing procedures do."
- 1999-02-13 R. Knauer: "Can that anti-skewing technique above generate all possible sequences including 111...1 and 000...0 with their expected probability based on their length?"
- 1999-02-13 karl malbrain: "What you're missing here is INDUCTION."
- 1999-02-15 Patrick Juola: "As to proof that the expected probability is based on the length; do it by case analysis and induction."
- 1999-02-15 Trevor Jackson, III: "In electronics I believe this is called 'edge detection' instead of 'level detection'."
- 1999-02-13 R. Knauer: "You may be able to anti-skew a sequence, but that won't remove correlation."
- 1999-02-15 Patrick Juola: "This method produces perfectly unbiased numbers *IF* the underlying bit sequence is independent."
- 1999-02-15 R. Knauer: "One advantage is that if the output is shorted or pulled up, there will be no final output since there will be no dibits of the kind 01 or 10."
- 1999-02-15 Patrick Juola: "...this is an engineering question, and not a formal requirement."
- 1999-02-15 R. Knauer: "Until another proveably secure anti-skewing technique can be identified, it has the advantage that it works."
- 1999-02-08 Terry Ritter: "...a 'bad' generator (one with a measurable error) will fail very badly virtually all of the time." "We seek to certify, to high probability, using statistical tests, that a complex physical process is producing the results we expect. In this way we certify the detection machinery, and then depend upon the assumed characteristics of the physical source as the generator of randomness."
- 1999-02-08 R. Knauer: "I do not understand what you have just said wrt the LSB's. Why would using larger character representations be preferably to using LSBs?"
- 1999-02-09 Terry Ritter: "Even when characters have an overall 'entropy' of less than a bit, there is no particular reason to expect to find the entropy concentrated in the LSB's. The LSB represents part of the coding; it has nothing to do with the entropy transported by the variation in usage of that coding."
- 1999-02-09 R. Knauer: "I suppose the problem I am having is that if you include all the bits of the text characters, you will be introducing bias into the stream."
- 1999-02-09 Mok-Kong Shen: "I believe lots of people would be very happy if you could tell them how to obtain a fair coin! Isn't evident now that one can never get an ideal OTP?"
- 1999-02-09 R. Knauer: "I meant that coin toss system as an analogy."
- 1999-02-09 Tony T. Warnock: "If you bounce a die hard enough, it acts random."
- 1999-02-08 Terry Ritter: "...as I see it, 'the' important quality for a cryptographic sequence is the independence of each value."
- 1999-01-29 Mok-Kong Shen: "...people doing statistical tests are very prudent and very modest ones. They never say anything categorically ('for sure', IS or IS NOT) but rather say 'something is fairly unlikely to be true' or 'there is not enough evidence to believe that it is false'."
- 1999-01-26 handWave: "If you are in the college dorm with enemies who may interfere remotely, then a thermal noise generator from an avalanche diode with a high gain amplifier breadboared using wire-wrapped, unshielded circuits is a bad implementation."
- 1999-01-26 Mok-Kong Shen: "Some one offers me two sequences generated from two different sources (I am not told which is which). I want to decide which one to use." "How am I going to make the decision?"
- 1999-01-26 Terry Ritter: "...as a very basic first step, we must be very sure that we actually are measuring and reporting molecular-level randomness, rather than some other signal." "Ideally, a randomness machine will allow us to conduct statistical experiments on the randomness source itself...."
- 1999-01-27 Mok-Kong Shen: "I like however to associate that 'known' with some degree of subjectivity (unless that 'known' can be shown to be the same as 'proved')"
- 1999-01-26 Trevor Jackson, III: "I doubt an average person could control a roulette wheel that well after a year of practice. OTOH, I think an average person could reach that level of skill with a silver dollar after a few days or a week."
- 1999-01-25 Medical Electronics Lab: "Using a smoke detector it's a mighty damn weak signal. It takes about 1e5 amplification to see the radiation noise and that involves lots of shielding from outside sources as well as preventing feedback."
- 1999-01-26 handWave: "Or take your smoke detector and place the vitals in contact with the surface of a modern DRAM: Large Random Signals would result in the big memory array."
- 1999-01-26 burt: "Have you done any Frequency domain analysis...?"
- 1999-01-26 handWave: "As I recall, the two faster oscillators ran at about 5Mhz to 50Mhz as their frequencies were randomly modulated."
- 1999-01-26 Terry Ritter: "I have never heard IC designers claim that it is 'not possible to tell' whether their design works. In fact, if we can't tell whether a design works or not, we generally consider that a design *failure*: If we can't test it, we can't trust it." "...it is in the nature of this sort of oscillator to synchronize to tiny transients as may occur from nearby state changes. So just how do we *know* that our oscillators *are* "unsynchronized"?" "I argue that an RNG as you describe not only has digital state, it also has analog thermal state, direct electrical coupling from other power-using circuits (including each stage), and indirect coupling from nearby circuits. Once we account for all this state and interaction, I expect that we can largely predict the result."
- 1999-01-26 Kazak, Boris: "Consider such a simple system... where HH is a... bottle, OO are Objects, MM is a Microphone. Now if we start rotating the Housing around its horizontal axis, the Objects will produce a loud Random Rattle, and the Microphone will transmit this rattle to the sound card."
- 1999-01-27 handWave: "Your sound card and software will need to comprehend the natural resonant frequencies of the bottle and filter out that repeating soundwave."
- 1999-01-27 R. Knauer: "I suspect that all classical processes, even chaotic ones, suffer from some kind of flaw as you describe. That's why I would only use quantum processes for a TRNG."
- 1999-01-27 handWave: "Your messages are beginning to repeat themselves, so I am not responding to these repeated comments."
- 1999-01-31 Bo Dömstedt: "As for random number generation, we sell them for $170...."
- 1999-01-31 Kazak, Boris: "In all the hype that is there on the WWW page you mentioned, there is not a single word about the underlying physical phenomena which produces 'high level signal random output'."
- 1999-02-01 Bo Dömstedt: "We use a noisy diode as our noise source. It has been extensively checked, and works well in practice."
- 1999-01-31 handWave: "You will be sipping your coffee while your lonely RNG will be under hostile attack by Martians."

```
Subject: hardRandNumbGen
Date: Fri, 22 Jan 1999 23:24:52 -1000
From: ".´¯`...´¯`...´¯`...´¯`...´¯`...´¯`.." &LT;fermion@false.net>
Message-ID: &LT;36A99564.D20@false.net>
Newsgroups: sci.crypt
Lines: 66
```

hardRandNumbGen

Those who seek to capture random numbers in their natural habitats often are faced with two difficult paths: to wait quietly for the small whispers of thermal noise or to become immersed in the dazzling thrashes of large signal sources. Both ways may lead to success, or to a failure so desperate, that every adversary may be seen as a stalker in the night. The story you are about to read is true: "The Hardware Random Number Generator!"

Our story starts in the late 1940's with a Bell Labs researcher in his Ivory Tower setting. Dusting off an old book on physics, a young researcher reads about noise from resistors:

> "Thermal noise is produced as a result of thermally excited random motion of free electrons in a conducting medium, such as a resistor".

This is the old wisdom of Kirchhoff and Kelvin. Randomness

may be gathered from an RMS voltage of about 4kTRB, where B is the bandwidth, R is the resistance, T is temperature, and k is Boltzman's constant. Faced with adversaries, the researcher knows he must use small resistors which are more immune to remote interference. But if 300 ohms are used, the voltage will only be two microvolts! An amplifier with a gain of a million will be needed to make the noise useable for his secret cryptographic purposes. Then the amplifier itself will become susceptible to outside influences.

Millions of people are depending on his team to find a better source of random numbers, when he has an inspiration. Like a collection of atoms whose motion so hard to predict, if he can use variable oscillators, or gyrators, as he likes to call them, then their combined signals would be hard to predict. Small variations in conditions would change the "large signal" outputs from his circuits, which he could sample at regular intervals. That was the beginning. Today, my friends, we are ready to receive the benefits of Large Signal Random Sources. No longer will we wait, with a hope and a prayer, that the microvolt sources of randomness will not fall victum to the beamed manipulations of deviant hackers, NO, digital large signals have brought us immunity from such a fate.

But it is not just the hacker who would mug our chaotic joy, it is the very regularity of our clock cycles and the very power of our conforming buses which threaten to impart a hideous regularity to our nonces, our IVs, our keys. The heartbeat of a computer is its clock, and a powerful hammerblow it is to any mere analog circuit which would dare to reside on our motherboards. This is why we cannot use sensitive amplifiers to boost the whispers of thermal noise. This is why Large Signal Sources are our refuge, our bounty, our provider of Hardware Random Number Generators. Oscillators, I tell you, OSCILLATORS, they are our main hope, and the pride modern civilization. I cannot exaggerate too much, the importance of avoiding the mistakes of past designers, who, through wishful thinking, risked it all, and lost, to the whims of a tiny hiss.

So go now, brash young designers of tomorrows crytosystems, go to your keyboards and your mice, and always remember: It is better to have thrashed and lost some quality, than to never have thrashed at all.

..´¯`...´¯`...´¯`...´¯`...´¯`...´¯`.. 1999/1/22

Subject: Re: hardRandNumbGen
Date: Sun, 24 Jan 1999 05:08:35 GMT
From: ritter@io.com (Terry Ritter)
Message-ID: <36aaaacf.5602532@news.io.com>
References: <36A99564.D20@false.net>
Newsgroups: sci.crypt
Lines: 53

On Fri, 22 Jan 1999 23:24:52 -1000, in <36A99564.D20@false.net>, in sci.crypt "..´¯`...´¯`...´¯`...´¯`...´¯`...´¯`.."
<fermion@false.net> wrote:

>[...]
>it is the very regularity of our clock cycles and the very
>power of our conforming buses which threaten to impart a
>hideous regularity to our nonces, our IVs, our keys. The
>heartbeat of a computer is its clock, and a powerful hammerblow
>it is to any mere analog circuit which would dare to reside
>on our motherboards. This is why we cannot use sensitive
>amplifiers to boost the whispers of thermal noise.

"Cannot" may be a bit of a stretch. Doing low-level analog in a digital environment is tough, there is no doubt about that. Presumably we would employ appropriate techniques to pull it off. Probably this would involve some form of power isolation and filtering, perhaps even shielding. Once the signal is large enough, it can compete with digital its own terms.

We note that the output of a CD player is supposed to be low-noise, yet is produced in a digital environment. And sound-card inputs amplify relatively low analog levels. Certainly, disk-drive magnetic read heads produce very low-level signals, yet are made to work well in a highly-digital environment.

>This is why
>Large Signal Sources are our refuge, our bounty, our provider
>of Hardware Random Number Generators. Oscillators, I tell you,
>OSCILLATORS, they are our main hope, and the pride modern
>civilization.

Unfortunately "oscillation" inherently seems to imply some amount of saved and time-delayed energy. It is this accumulation of energy that makes it difficult to change the oscillation, and that is normally an advantage. Normally, an oscillator cannot detect quantum or molecular phenomena, and we would not want it to do so.

A signal composed of many oscillators, each doing their own thing, is
admittedly complex. But complex relationships are not, by themselves,
cryptographically secure. We could even think to simulate such a
system numerically, in which case the system is clearly no more than
yet another pseudorandom state machine waiting to be exposed. And
while any such simulation might not be exact, it could be close, and
we could continually adjust the simulation to the reality we do see.

Terry Ritter ritter@io.com http://www.io.com/~ritter/
Crypto Glossary http://www.io.com/~ritter/GLOSSARY.HTM

Subject: Re: hardRandNumbGen
Date: Sun, 24 Jan 1999 03:53:40 -1000
From: ".·´¯`···´¯`···´¯`···´¯`···´¯`···´¯`··" <real@complex.net>
Message-ID: <36AB25E4.2E7E@complex.net>
References: <36aaaacf.5602532@news.io.com>
Newsgroups: sci.crypt
Lines: 130

Terry Ritter wrote:
>
> On Fri, 22 Jan 1999 23:24:52 -1000, in <36A99564.D20@false.net>, in
> sci.crypt ".·´¯`···´¯`···´¯`···´¯`···´¯`···´¯`··"
> <fermion@false.net> wrote:
>
> >[...]
> >it is the very regularity of our clock cycles and the very
> >power of our conforming buses which threaten to impart a
> >hideous regularity to our nonces, our IVs, our keys. The
> >heartbeat of a computer is its clock, and a powerful hammerblow
> >it is to any mere analog circuit which would dare to reside
> >on our motherboards. This is why we cannot use sensitive
> >amplifiers to boost the whispers of thermal noise.

>"Cannot" may be a bit of a stretch.

Yes, my prose were intended to be didactic to troll for responses.
Thank you for your polite and rational response.

>Doing low-level analog in a
>digital environment is tough, there is no doubt about that.
>Presumably we would employ appropriate techniques to pull it off.
>Probably this would involve some form of power isolation and
>filtering, perhaps even shielding. Once the signal is large enough,
>it can compete with digital its own terms.

On a single chip product, like a mainstream microprocessor that is
employing appropriate techniques to push the limits of speed, you
may find that large signals for a random number generator are
preferable to millivolt signals. The substrate junction with p+ and
n- wells have a capacitive noise for which it is hard to provide
accurate cancellation.

>We note that the output of a CD player is supposed to be low-noise,
>yet is produced in a digital environment. And sound-card inputs
>amplify relatively low analog levels. Certainly, disk-drive magnetic
>read heads produce very low-level signals, yet are made to work well
>in a highly-digital environment.

The inputs to a CD player are large signal, digital inputs from light
relections. The digital codes are reproduced from a recording studio
which spent millions to get way from periodic noise. I have not done
the following experiment: put a spectrum analyser on the output of a CD
player during quiet passages. Look for the noise outside of the human
hearing range. I expect that the digital electronics in a CD player
produce ordinary levels of periodic noise that we cannot hear. And
that includes non-random noise BELOW 40 hz.

CD players are not 400 Mhz microprocessors that are going as fast
as possible driving motherboard capacitances on 400 pins. They are
slow, dedicated chips with few pins being driven, and with small
load capacitances. They are self-contained assemblies that are
shielded from other components in a home stereo system. The kind of
RNG I am interested in is one that is robust. One that is prepared
to exist in a hostile electrical environment, not some pampered
little dog of a processor.

Hard drives are limited to 5 zeros in a row. Consider : Why?

>>This is why
>>Large Signal Sources are our refuge, our bounty, our provider
>>of Hardware Random Number Generators. Oscillators, I tell you,
>>OSCILLATORS, they are our main hope, and the pride of modern
>>civilization.

>Unfortunately "oscillation" inherently seems to imply some amount of
>saved and time-delayed energy. It is this accumulation of energy that
>makes it difficult to change the oscillation, and that is normally an
>advantage. Normally, an oscillator cannot detect quantum or molecular
>phenomena, and we would not want it to do so.

A digital ring oscillator composed of Schmitt Triggers (with hysteresis) can be designed to have a slow rise time of ten microseconds, but they respond to an input transition in 100ps, to use round numbers. To illustrate the powerful effect that these facts have on the recording of thermal noise, I will give the details of it operation. Assume there is a +/- 1 millivolt thermal noise present on the output of an inverter. Assume a Schmitt trigger will switch when its input rises above 1v for a system using 2v power supplies. The oscillator runs at 100khz. How long will it take for the input to rise 2mV? That is 2mV divided by 1V/10us or 50ns. So on every cycle of the oscillator there is a 50ns time when uncertainty exists. This is a half percent on each cycle.

Since multiple oscillators will be involved, each with a half percent uncertainty, one can see that by using 200 such oscillators XORed together, the output would be quite random. But in practice, 200 oscillators are not needed because there are several sources of uncertainty in a well designed Large Signal Random Number Generator such as the one I designed at a large semiconductor company. It was fabricated and tested by a team of engineers. It worked well. It was evaluated by the CIA and they sent us a report on its characteristics. Have you built any hardware random number generators using large signals? Small signals?

Mr. Ritter says, "It is this accumulation of energy that makes it difficult to change the oscillation....". It is easy to change the oscillation period using capacitors that are connected or disconnected from the oscillator by switches that are controlled by signals from the random string.
This arrangement amplifies any thermal noise that is captured.
To be more detailed, when a 1mV noise does affect a bit value that is shifted into a register, that bit value changes the frequency of oscillation of one or more oscillators. The XOR combines these changes for a while until the combined bitstream is sampled. I contend that this is not just illusory complexity, it is an amplification of a thermal noise into a large product.

>A signal composed of many oscillators, each doing their own thing, is
>admittedly complex. But complex relationships are not, by themselves,
>cryptographically secure. We could even think to simulate such a
>system numerically, in which case the system is clearly no more than
>yet another pseudorandom state machine waiting to be exposed. And
>while any such simulation might not be exact, it could be close, and
>we could continually adjust the simulation to the reality we do see.

I hope that you would simulate the 200 oscillator example I gave. Yes, you can add the +/- 1mV noise and adjust to make it as accurate as you care to invest time for. I have read your pat statement above several times recently, and I disagree with it. It is possible to design a complex circuit that is poorly done and which therefore would fail tests for randomness. But you should not get hung up on poor designs that fit your expectation. You should open your mind to the possibility that talented design engineers might do a good job using techniques you wish did not exist. You can change your opinion.
I will send you a relevent patent number through private email so you can see the drawings.

..´¯`...´¯`...´¯`...´¯`...´¯`...´¯`..

Subject: Re: hardRandNumbGen
Date: Sun, 24 Jan 1999 17:48:24 GMT
From: ritter@io.com (Terry Ritter)
Message-ID: <36ab5cde.5814163@news.io.com>
References: <36AB25E4.2E7E@complex.net>
Newsgroups: sci.crypt
Lines: 171

On Sun, 24 Jan 1999 03:53:40 -1000, in <36AB25E4.2E7E@complex.net>, in sci.crypt "..´¯`...´¯`...´¯`...´¯`...´¯`...´¯`.."
<real@complex.net> wrote:

>Terry Ritter wrote:
>[...]
>>Unfortunately "oscillation" inherently seems to imply some amount of
>>saved and time-delayed energy. It is this accumulation of energy that
>>makes it difficult to change the oscillation, and that is normally an
>>advantage. Normally, an oscillator cannot detect quantum or molecular
>>phenomena, and we would not want it to do so.
>
>A digital ring oscillator composed of Schmitt Triggers (with
>hysteresis) can be designed to have a slow rise time of ten
>microseconds,

Presumably this means that the effective source resistance is large compared to the input capacitance, and the source of the delay is an R-C ramp to the next stage.

>but they respond to an input transition in 100ps, to
>use round numbers. To illustrate the powerful effect that these
>facts have on the recording of thermal noise, I will give the
>details of it operation. Assume there is a +/- 1 millivolt thermal

>noise present on the output of an inverter.

That means that this "large signal" design is probably sensitive to even tiny power and ground transients. It is going to be very hard to distinguish the effects of "real" thermal noise from transient feedback due to the structure of the circuit. So how can we have confidence in the result? Statistical testing cannot distinguish between "physical" and "pseudo" randomness.

>Assume a Schmitt trigger
>will switch when its input rises above 1v for a system using 2v
>power supplies. The oscillator runs at 100khz. How long will it take
>for the input to rise 2mV? That is 2mV divided by 1V/10us or 50ns.
>So on every cycle of the oscillator there is a 50ns time when
>uncertainty exists. This is a half percent on each cycle.

As a rare peak value, presumably.

>Since multiple oscillators will be involved, each with a half percent
>uncertainty, one can see that by using 200 such oscillators XORed
>together, the output would be quite random.

What I see is a huge complexity-based increase in signal transitions (a frequency increase) which will be hard to distinguish from heat-based noise. And if we cannot distinguish operation *with* noise from operation *without* noise, we have no way to prove that noise is involved at all. Other than claims and handwaves, of course.

>But in practice, 200
>oscillators are not needed because there are several sources of
>uncertainty in a well designed Large Signal Random Number Generator
>such as the one I designed at a large semiconductor company. It was
>fabricated and tested by a team of engineers. It worked well.

I have looked at the published results a number of times. They were in fact part of the basis for my investigation of the numerical relationship between repeats in sampling and the overall population.

Easy calculations using the publushed results show that the effective population of values is 1/4 the claimed ideal, which shows that the design was not as good as you thought.

>It was
>evaluated by the CIA and they sent us a report on its
>characteristics.

The published (admittedly meager) experimental evidence says otherwise.

>Have you built any hardware random number
>generators using large signals?

The claimed basis for your generator is thermal noise, which is NOT large-signal. A large-signal digital system is a PSEUDO-random digital RNG, and can be implemented in software as well as hardware. So, yes, certainly I have implemented and tested many large signal (software) RNG's.

Some software computations are hard to reverse. But few if any of the conventional statistical RNG's have stood up to attack. Just giving a hardware design and claiming "nobody can break this" is the sort of thing we see on sci.crypt all the time.

The reasoning about this design is contradictory: Supposedly the large signal design is "random" because it senses low-level noise. Yet the circuit is supposedly suitable for a noisy digital chip because it is a "large-signal" design. There is a fundamental problem in making both claims at the same time.

>Small signals?

Yes.

>Mr. Ritter says, "It is this accumulation of energy that makes it
>difficult to change the oscillation...CRYPHTML.HTM". It is easy to change the
>oscillation period using capacitors that are connected or
>disconnected from the oscillator by switches that are
>controlled by signals from the random string.

But that approach is digital complexity, and not thermal randomness. It can be simulated in software. It is PSEUDO-random. Maybe it is strong, maybe not, but there is certainly no proof.

>This arrangement amplifies any thermal noise that is captured.
>To be more detailed, when a 1mV noise does affect a bit value that
>is shifted into a register, that bit value changes the frequency
>of oscillation of one or more oscillators. The XOR combines these
>changes for a while until the combined bitstream is sampled. I
>contend that this is not just illusory complexity, it is an
>amplification of a thermal noise into a large product.

The obvious experiment, then, is to take the device to cryogenic temperatures and see how it performs. If the output still has good statistics, we can suspect that the output does not represent thermal noise at all, but is just a complex digital system. Was such an

experiment performed?

>>A signal composed of many oscillators, each doing their own thing, is
>>admittedly complex. But complex relationships are not, by themselves,
>>cryptographically secure. We could even think to simulate such a
>>system numerically, in which case the system is clearly no more than
>>yet another pseudorandom state machine waiting to be exposed. And
>>while any such simulation might not be exact, it could be close, and
>>we could continually adjust the simulation to the reality we do see.
>
>I hope that you would simulate the 200 oscillator example I gave.

But your design does not use 200 oscillators, does it?

>Yes,
>you can add the +/- 1mV noise and adjust to make it as accurate as you
>care to invest time for. I have read your pat statement above several
>times recently, and I disagree with it. It is possible to design a
>complex circuit that is poorly done and which therefore would fail
>tests for randomness.

Even PSEUDO-random RNG's pass statistical tests. Those tests have
nothing to do with cryptographic unpredictability or "strength." Yet
strength is what you claim.

>But you should not get hung up on poor designs
>that fit your expectation. You should open your mind to the
>possibility that talented design engineers might do a good job using
>techniques you wish did not exist.

I have no such wish.

>You can change your opinion.

I think you have missed the distinction between unpredictable
randomness for cryptography, and ordinary statistical randomness.

>I will send you a relevent patent number through private email so
>you can see the drawings.

I have seen the technical article.

Terry Ritter ritter@io.com http://www.io.com/~ritter/
Crypto Glossary http://www.io.com/~ritter/GLOSSARY.HTM

Subject: Re: hardRandNumbGen
Date: Sun, 24 Jan 1999 18:45:25 GMT
From: rcktexas@ix.netcom.com (R. Knauer)
Message-ID: <36ab69fd.16112067@nntp.ix.netcom.com>
References: <36ab5cde.5814163@news.io.com>
Newsgroups: sci.crypt
Lines: 17

On Sun, 24 Jan 1999 17:48:24 GMT, ritter@io.com (Terry Ritter) wrote:

>Even PSEUDO-random RNG's pass statistical tests. Those tests have
>nothing to do with cryptographic unpredictability or "strength."

That statement needs to be added to the FAQ on Crypto-Grade
Randomness.

It says it all.

Bob Knauer

"It is not the function of our government to keep the citizen from
falling into error; it is the function of the citizen to keep the
government from falling into error."
--Justice Robert H. Jackson

Subject: Re: hardRandNumbGen
Date: 25 Jan 99 02:37:29 GMT
From: jsavard@ecn.ab.ca ()
Message-ID: <36abd8e9.0@ecn.ab.ca>
References: <36ab69fd.16112067@nntp.ix.netcom.com>
Newsgroups: sci.crypt
Lines: 17

R. Knauer (rcktexas@ix.netcom.com) wrote:
: On Sun, 24 Jan 1999 17:48:24 GMT, ritter@io.com (Terry Ritter) wrote:

: >Even PSEUDO-random RNG's pass statistical tests. Those tests have
: >nothing to do with cryptographic unpredictability or "strength."

: That statement needs to be added to the FAQ on Crypto-Grade

: Randomness.

: It says it all.

It does indeed, but it will probably have to be expanded and commented
upon before it will "say it all" clearly enough so that everyone
understands what it means. Many people have heard this, but because they
have not understood, they did not believe.

John Savard

Subject: Re: hardRandNumbGen
Date: Mon, 25 Jan 1999 11:55:36 GMT
From: rcktexas@ix.netcom.com (R. Knauer)
Message-ID: <36ac5b28.1691712@nntp.ix.netcom.com>
References: <36abd8e9.0@ecn.ab.ca>
Newsgroups: sci.crypt
Lines: 73

On 25 Jan 99 02:37:29 GMT, jsavard@ecn.ab.ca () wrote:

>: >Even PSEUDO-random RNG's pass statistical tests. Those tests have
>: >nothing to do with cryptographic unpredictability or "strength."

>: That statement needs to be added to the FAQ on Crypto-Grade
>: Randomness.
>: It says it all.

>It does indeed, but it will probably have to be expanded and commented
>upon before it will "say it all" clearly enough so that everyone
>understands what it means. Many people have heard this, but because they
>have not understood, they did not believe.

I agree. Here is a post from Patrick Juola that expands on this in a
way that can be understood by all.

+++++
On 21 Jan 1999 08:23:54 -0500, juola@mathcs.duq.edu (Patrick Juola)
wrote:

You're not seeing the fundamnental distinction between "irrationality"
and "randomness" in that randomness is a function, not of a number,
but of a process.

Just for clarification : *Any* number/string can be the result of
a uniformly random process. In fact, a uniformly random process will
always produce all numbers equiprobably, by construction.

Any number can also be produced as the result of a non-random process,
although for many numbers this will be a very uninteresting process
such as a simple table-lookup and copy.

The closest relative for irrationality is not the properties such
as "non-repeating fraction" (which is a thoroughly bogus definition,
by the way), but the method by which you GET a rational number.

To wit, a rational number can be generated as the ratio of two
integers p and q (q != 0 for the formalists, pthththththth). An
irrational number is a number that cannot be so generated.

Now, it so happens (lucky us) that any number that can be generated
as the ratio of two integers can also be written as a terminating
and/or repeating continued decimal string. This is an independent
property, first proved in the year <mumble> by someone no doubt too
famous for me to remember offhand. But the fact that you can
characterize a number as rational or irrational by inspection is,
strictly speaking, a lucky fluke.

There's a similar definition for, e.g., transcendentals -- a
transcendental number, of course, is a number that cannot be produced
as the solution to a polynomial equation. Transcendentals are a
strict subset of irrationals -- sqrt(2), for instance, is irrational
but not transcendental. However, there's no way to characterize *by
inspection* whether or not a given irrational number is
transcendental. I can easily prove a given number is *NOT*
transcendental by showing a polynomial to which &c., but
I can't go the other way.

So the point is that the characterization of both irrationals and
transcendentals is a) strictly process-driven, and b) defined in the
negative sense -- "no possible way to...CRYPHTML.HTM" That irrationals can be
cleanly defined in typographic properties should *not* lead you to
believe that randomness can also be defined in typographic
properties or that it can be defined in positive terms.
+++++

Bob Knauer

"An honest man can feel no pleasure in the exercise of power over
his fellow citizens."
--Thomas Jefferson

Subject: Re: hardRandNumbGen
Date: Mon, 25 Jan 1999 04:44:51 -1000
From: ".´¯`...´¯`...´¯`...´¯`...´¯`...´¯`.." <real@complex.net>
Message-ID: <36AC8363.6D55@complex.net>
References: <36abd8e9.0@ecn.ab.ca>
Newsgroups: sci.crypt
Lines: 240

On Sun, 24 Jan 1999 03:53:40 -1000, in <36AB25E4.2E7E@complex.net>, in sci.crypt <real@complex.net> sinewave wrote:

>Terry Ritter wrote:

sinewave:
>>A digital ring oscillator composed of Schmitt Triggers (with
>>hysteresis) can be designed to have a slow rise time of ten
>>microseconds,

Terry:
>Presumably this means that the effective source resistance is large
>compared to the input capacitance, and the source of the delay is an
>R-C ramp to the next stage.

Yes.

>>but they respond to an input transition in 100ps, to
>>use round numbers. To illustrate the powerful effect that these
>>facts have on the recording of thermal noise, I will give the
>>details of it operation. Assume there is a +/- 1 millivolt thermal
>>noise present on the output of an inverter.

>That means that this "large signal" design is probably sensitive to
>even tiny power and ground transients. It is going to be very hard to
>distinguish the effects of "real" thermal noise from transient
>feedback due to the structure of the circuit. So how can we have
>confidence in the result? Statistical testing cannot distinguish
>between "physical" and "pseudo" randomness.

In the real world, it is not always possible to tell. Integrating
a random number generator (RNG) on a commodity IC is similar to
a manned expedition to MARS: they must take everything with them
into that harsh environment that they will need. If the craft is
buffeted by periodic winds, they do not have the luxury of calling
back to base and saying, "Houston, you told us this was a vacuum,
please make it a perfect vacuum, over". The RNG will encounter
non-ideal electrical environments. It should have redundant systems
which are combined to give the final random number the best shot at
being unpredictable, not perfect, but unpredictable. The multiple
ring oscillator design described here should be a part of the
on-chip subsystem: it is an unpredictable seed generator. One can
also add a differential amplifier RNG with power supply noise
rejection capabilities, a PRNG, a counter, a hash, and storage for
the previous random number to use for initialization and for
checking firmware. The RNG described above is a Large Signal
Random Number Generator, to be described in more detail, below.

>>Assume a Schmitt trigger
>>will switch when its input rises above 1v for a system using 2v
>>power supplies. The oscillator runs at 100khz. How long will it take
>>for the input to rise 2mV? That is 2mV divided by 1V/10us or 50ns.
>>So on every cycle of the oscillator there is a 50ns time when
>>uncertainty exists. This is a half percent on each cycle.

>As a rare peak value, presumably.

Yes, I am using round numbers.

>>Since multiple oscillators will be involved, each with a half percent
>>uncertainty, one can see that by using 200 such oscillators XORed
>>together, the output would be quite random.

>What I see is a huge complexity-based increase in signal transitions
>(a frequency increase) which will be hard to distinguish from
>heat-based noise. And if we cannot distinguish operation *with* noise
>from operation *without* noise, we have no way to prove that noise is
>involved at all. Other than claims and handwaves, of course.

I am glad you raised the "handwaves" metaphore, because handwaves are
what toss coins. A complex person tosses a coin and you might think it
is random. The oscillator RNG in this discussion is directly analogous
to a coin toss in many ways. If a coin is not rotating (oscillating)
it will fall back into the hand in the same position that it stated from.
It is the rotation that helps the randomness, not only the complexity
of the nervous system, the elasticity of the skin, and the trembling of
the muscles. The rotation should be fast for best results. A juggler
could become skilled at non-random coin tosses for one coin that
rotates slowly. But if she tosses three coins with rapid rotation than
it is likely that random results will occur. If a periodic wind is
present in a coin toss, yes, it will influence the outcome, but the
result will often be recognizable as a useful random throw, or a throw
that was blown away. The same with this RNG.

The major source of randomness of this RNG is the unsynchronized
nature of multiple oscillators with randomly changing frequencies. This

is a large signal phenomenon, which cannot be accurately described mathematically. Similar to a coin toss, many analog variables are involved. These continuous variations of many influences cause seeming randomness. If you can mathematically describe a human coin toss, then so you can with this RNG. But you cannot, and I cannot. That does not invalidate the usefulness of these seed generators, not in this century.

>>But in practice, 200
>>oscillators are not needed because there are several sources of
>>uncertainty in a well designed Large Signal Random Number Generator
>>such as the one I designed at a large semiconductor company. It was
>>fabricated and tested by a team of engineers. It worked well.

>I have looked at the published results a number of times. They were
>in fact part of the basis for my investigation of the numerical
>relationship between repeats in sampling and the overall population.

>Easy calculations using the publushed results show that the effective
>population of values is 1/4 the claimed ideal, which shows that the
>design was not as good as you thought.

Correct, that first version in that report had an XOR gate placed in a bad position, causing twice as many ones as zeros. The CIA alerted us to my mistake with that one gate. When removed, the results are much better. I still regret my mistake in that one gate placement.

>>It was
>>evaluated by the CIA and they sent us a report on its
>>characteristics.

>The published (admittedly meager) experimental evidence says
>otherwise.

Single reports do not tell all of the facts.

>>Have you built any hardware random number
>>generators using large signals?

>The claimed basis for your generator is thermal noise, which is NOT
>large-signal. A large-signal digital system is a PSEUDO-random
>digital RNG, and can be implemented in software as well as hardware.
>So, yes, certainly I have implemented and tested many large signal
>(software) RNG's.

The ealier description was an illustration for some readers to examine. It was not an exhaustive explanation of the theory behind the design. I have now expanded upon the description, explaining the large signals as being analogous to coin tosses which must rotate due to a complex had waving motion. The complexity of my circuit design mimics, on a small scale, the complexities of the human hand wave and coin toss. The frequency changes in the design are the analogy of the hand motion. Thermal irregularities power supply variations also contribute to this hand motion.

Radioactive decay is also a large signal RNG. It may be considered to be both digital and analog, as this RNG may be.

>Some software computations are hard to reverse. But few if any of the
>conventional statistical RNG's have stood up to attack. Just giving a
>hardware design and claiming "nobody can break this" is the sort of
>thing we see on sci.crypt all the time.

I do not claim nobody can break this. I am presenting concepts to a wide reading audience. Some of these concepts are less sound than others, so the readers have the opportunity to judge various attepts to produce randomness in a harsh environment. I hope that they will fare better than I did.

>The reasoning about this design is contradictory: Supposedly the
>large signal design is "random" because it senses low-level noise.
>Yet the circuit is supposedly suitable for a noisy digital chip
>because it is a "large-signal" design. There is a fundamental problem
>in making both claims at the same time.

I have addressed this above. A large signal, digital oscillator has small noise on top of that. The randomness is primarily based on the coin toss analogy. The thermal noise calculation first given is a secondary source of randomness. The periodic power supply noise will affect this design more in some ways than it would affect an analog circuit with well designed differential and common mode considerations. But the ways periodic noise affects these circuits do not ruin the unpredictability of the resulting numbers. I leave that discussion for another day.

snip...

>>Mr. Ritter says, "It is this accumulation of energy that makes it
>>difficult to change the oscillation...CRYPHTML.HTM". It is easy to change the
>>oscillation period using capacitors that are connected or
>>disconnected from the oscillator by switches that are
>>controlled by signals from the random string.

>But that approach is digital complexity, and not thermal randomness.
>It can be simulated in software. It is PSEUDO-random. Maybe it is

>strong, maybe not, but there is certainly no proof.

It is analog complexity. I will give no proof today. Give me proof
of coin tossing that does not involve complexity or strength..

snip..

>The obvious experiment, then, is to take the device to cryogenic
>temperatures and see how it performs. If the output still has good
>statistics, we can suspect that the output does not represent thermal
>noise at all, but is just a complex digital system. Was such an
>experiment performed?

No. The circuit depends on many complex factors for randomness, as a
coin toss does. In some imagined laboratory experiment, it is feasible
to control all factors, causing non-random results. In commodity
applications, Large Signal Random Number Generators are sometimes
superior to small signal based generators and both may appear on a
single IC.

>>>A signal composed of many oscillators, each doing their own thing, is
>>>admittedly complex. But complex relationships are not, by themselves,
>>>cryptographically secure. We could even think to simulate such a
>>>system numerically, in which case the system is clearly no more than
>>>yet another pseudorandom state machine waiting to be exposed. And
>>>while any such simulation might not be exact, it could be close, and
>>>we could continually adjust the simulation to the reality we do see.
>>
>>I hope that you would simulate the 200 oscillator example I gave.

>But your design does not use 200 oscillators, does it?

No it had 3. The 200 oscillator example is for a simplified explanation
of one source of randomness.

>>Yes,
>>you can add the +/- 1mV noise and adjust to make it as accurate as you
>>care to invest time for. I have read your pat statement above several
>>times recently, and I disagree with it. It is possible to design a
>>complex circuit that is poorly done and which therefore would fail
>>tests for randomness.

>Even PSEUDO-random RNG's pass statistical tests. Those tests have
>nothing to do with cryptographic unpredictability or "strength." Yet
>strength is what you claim.

Yes it is a strong source, as upcoming product releases are expected to
show. Just because old PRNGs pass some tests does not mean that new
designs are bad, as you imply.

>I think you have missed the distinction between unpredictable
>randomness for cryptography, and ordinary statistical randomness.

A PSRG may be depended upon to produce the same string under certain
easy to arrange conditions. This RNG does the opposite of that. Two
sequential random numbers from this circuit would prove that to
anyone who tests it, most of the time.

Thank you for this polite discussion.

Subject: Re: hardRandNumbGen
Date: Mon, 25 Jan 1999 13:23:11 -0500
From: "Trevor Jackson, III" <fullmoon@aspi.net>
Message-ID: <36ACB68D.34FA34A4@aspi.net>
References: <36AC8363.6D55@complex.net>
Newsgroups: sci.crypt
Lines: 290

Two points, in context below...

..´¯`...´¯`...´¯`...´¯`...´¯`...´¯`.. wrote:

> On Sun, 24 Jan 1999 03:53:40 -1000, in <36AB25E4.2E7E@complex.net>, in
> sci.crypt <real@complex.net> sinewave wrote:
>
> >Terry Ritter wrote:
>
> sinewave:
> >>A digital ring oscillator composed of Schmitt Triggers (with
> >>hysteresis) can be designed to have a slow rise time of ten
> >>microseconds,
>
> Terry:
> >Presumably this means that the effective source resistance is large
> >compared to the input capacitance, and the source of the delay is an
> >R-C ramp to the next stage.
>
> Yes.
>
> >>but they respond to an input transition in 100ps, to
> >>use round numbers. To illustrate the powerful effect that these
> >>facts have on the recording of thermal noise, I will give the
> >>details of it operation. Assume there is a +/- 1 millivolt thermal

> >>noise present on the output of an inverter.
>
> >That means that this "large signal" design is probably sensitive to
> >even tiny power and ground transients. It is going to be very hard to
> >distinguish the effects of "real" thermal noise from transient
> >feedback due to the structure of the circuit. So how can we have
> >confidence in the result? Statistical testing cannot distinguish
> >between "physical" and "pseudo" randomness.
>
> In the real world, it is not always possible to tell. Integrating
> a random number generator (RNG) on a commodity IC is similar to
> a manned expedition to MARS: they must take everything with them
> into that harsh environment that they will need. If the craft is
> buffeted by periodic winds, they do not have the luxury of calling
> back to base and saying, "Houston, you told us this was a vacuum,
> please make it a perfect vacuum, over". The RNG will encounter
> non-ideal electrical environments. It should have redundant systems
> which are combined to give the final random number the best shot at
> being unpredictable, not perfect, but unpredictable. The multiple
> ring oscillator design described here should be a part of the
> on-chip subsystem: it is an unpredictable seed generator. One can
> also add a differential amplifier RNG with power supply noise
> rejection capabilities, a PRNG, a counter, a hash, and storage for
> the previous random number to use for initialization and for
> checking firmware. The RNG described above is a Large Signal
> Random Number Generator, to be described in more detail, below.
>
> >>Assume a Schmitt trigger
> >>will switch when its input rises above 1v for a system using 2v
> >>power supplies. The oscillator runs at 100khz. How long will it take
> >>for the input to rise 2mV? That is 2mV divided by 1V/10us or 50ns.
> >>So on every cycle of the oscillator there is a 50ns time when
> >>uncertainty exists. This is a half percent on each cycle.
>
> >As a rare peak value, presumably.
>
> Yes, I am using round numbers.
>
> >>Since multiple oscillators will be involved, each with a half percent
> >>uncertainty, one can see that by using 200 such oscillators XORed
> >>together, the output would be quite random.
>
> >What I see is a huge complexity-based increase in signal transitions
> >(a frequency increase) which will be hard to distinguish from
> >heat-based noise. And if we cannot distinguish operation *with* noise
> >from operation *without* noise, we have no way to prove that noise is
> >involved at all. Other than claims and handwaves, of course.
>
> I am glad you raised the "handwaves" metaphore, because handwaves are
> what toss coins. A complex person tosses a coin and you might think it
> is random. The oscillator RNG in this discussion is directly analogous
> to a coin toss in many ways. If a coin is not rotating (oscillating)
> it will fall back into the hand in the same position that it stated from.
> It is the rotation that helps the randomness, not only the complexity
> of the nervous system, the elasticity of the skin, and the trembling of
> the muscles. The rotation should be fast for best results. A juggler
> could become skilled at non-random coin tosses for one coin that
> rotates slowly. But if she tosses three coins with rapid rotation than
> it is likely that random results will occur. If a periodic wind is
> present in a coin toss, yes, it will influence the outcome, but the
> result will often be recognizable as a useful random throw, or a throw
> that was blown away. The same with this RNG.

Human gestures are not a good foundation for system design. There are large, industrial concerns that rely upon human-gesture-generated unpredictability. Their interest is *not* statistical randomness as we find in simulations, games, and Monte Carlo tests (in spite of the latter name). Their interest is the same as ours: unpredicability. They are called casinos.

In spite of the fantastic efforts taken to eliminate predictability in games of chance human gestures can still dominate the outcomes completely. I'm not referring to shuffling cards systematically, but to rolling a roulette ball against the wheel so precisely that out of 20 tries a human can obtain a predicted outcome (slot 17) 10 times. 50% success. I think that constitutes predictability.

The hand-eye coordination involved is of an extreme level, requiring decades of practice to achieve. But it is real. The complexity of controlling a roulette wheel appears to me to be far larger than that of a coin toss. Even a fast one.

Without detailed scrutiny of your design I cannot tell whether it is robust. No inspection of the outcome will convince me it is. However, the design philosophy you have expressed leads me to believe there will be weaknesses in the system.

> The major source of randomness of this RNG is the unsynchronized
> nature of multiple oscillators with randomly changing frequencies. This
> is a large signal phenomenon, which cannot be accurately described
> mathematically. Similar to a coin toss, many analog variables are
> involved. These continuous variations of many influences cause seeming
> randomness. If you can mathematically describe a human coin toss, then
> so you can with this RNG. But you cannot, and I cannot. That does not
> invalidate the usefulness of these seed generators, not in this
> century.

The phrase "randomly changing oscillators" is key to the paragrph above. I would like to question the use of the term random in the sense of unpredictable. Since the (intermediate) output of the system is driving the changes to the oscillators there is a full feedback loop present. This kind of system may pass statistical tests for randomness, but it may not be unpredictable. The result may "cause seeming randomness", but this is far from unpredictability. For instance, how much correlation would you expect from a set of such devices initialized identically?

Even the presmption that the output would pass statistical tests is questionable. One famous gafffe in PRNG design was Knuth's composite generator, which he called superrandom. Unfortunately it was a closed loop design. He did not forsee the possibility of cycles so short as to be degenerate. All closed loop designs contain this danger. If the hardware output is driving the hardware configuration, it is avidly searching for a configuration that represents a local minima in its volatility.

Now, given initialization in a configuration near such a local minima, how much divergence would we find in the output of a set of these devices?

> >>But in practice, 200
> >>oscillators are not needed because there are several sources of
> >>uncertainty in a well designed Large Signal Random Number Generator
> >>such as the one I designed at a large semiconductor company. It was
> >>fabricated and tested by a team of engineers. It worked well.
>
> >I have looked at the published results a number of times. They were
> >in fact part of the basis for my investigation of the numerical
> >relationship between repeats in sampling and the overall population.
>
> >Easy calculations using the publushed results show that the effective
> >population of values is 1/4 the claimed ideal, which shows that the
> >design was not as good as you thought.
>
> Correct, that first version in that report had an XOR gate placed in
> a bad position, causing twice as many ones as zeros. The CIA alerted
> us to my mistake with that one gate. When removed, the results are
> much better. I still regret my mistake in that one gate placement.
>
> >>It was
> >>evaluated by the CIA and they sent us a report on its
> >>characteristics.
>
> >The published (admittedly meager) experimental evidence says
> >otherwise.
>
> Single reports do not tell all of the facts.
>
> >>Have you built any hardware random number
> >>generators using large signals?
>
> >The claimed basis for your generator is thermal noise, which is NOT
> >large-signal. A large-signal digital system is a PSEUDO-random
> >digital RNG, and can be implemented in software as well as hardware.
> >So, yes, certainly I have implemented and tested many large signal
> >(software) RNG's.
>
> The ealier description was an illustration for some readers to examine.
> It was not an exhaustive explanation of the theory behind the design.
> I have now expanded upon the description, explaining the large
> signals as being analogous to coin tosses which must rotate due to
> a complex had waving motion. The complexity of my circuit design
> mimics, on a small scale, the complexities of the human hand wave
> and coin toss. The frequency changes in the design are the analogy
> of the hand motion. Thermal irregularities power supply variations
> also contribute to this hand motion.
>
> Radioactive decay is also a large signal RNG. It may be considered
> to be both digital and analog, as this RNG may be.
>
> >Some software computations are hard to reverse. But few if any of the
> >conventional statistical RNG's have stood up to attack. Just giving a
> >hardware design and claiming "nobody can break this" is the sort of
> >thing we see on sci.crypt all the time.
>
> I do not claim nobody can break this. I am presenting concepts to a
> wide reading audience. Some of these concepts are less sound than
> others, so the readers have the opportunity to judge various attepts
> to produce randomness in a harsh environment. I hope that they will
> fare better than I did.
>
> >The reasoning about this design is contradictory: Supposedly the
> >large signal design is "random" because it senses low-level noise.
> >Yet the circuit is supposedly suitable for a noisy digital chip
> >because it is a "large-signal" design. There is a fundamental problem
> >in making both claims at the same time.
>
> I have addressed this above. A large signal, digital oscillator has
> small noise on top of that. The randomness is primarily based on the
> coin toss analogy. The thermal noise calculation first given is a
> secondary source of randomness. The periodic power supply noise
> will affect this design more in some ways than it would affect an
> analog circuit with well designed differential and common mode
> considerations. But the ways periodic noise affects these circuits

> do not ruin the unpredictability of the resulting numbers. I leave
> that discussion for another day.
>
> snip...
>
> >>Mr. Ritter says, "It is this accumulation of energy that makes it
> >>difficult to change the oscillation...CRYPHTML.HTM". It is easy to change the
> >>oscillation period using capacitors that are connected or
> >>disconnected from the oscillator by switches that are
> >>controlled by signals from the random string.
>
> >But that approach is digital complexity, and not thermal randomness.
> >It can be simulated in software. It is PSEUDO-random. Maybe it is
> >strong, maybe not, but there is certainly no proof.
>
> It is analog complexity. I will give no proof today. Give me proof
> of coin tossing that does not involve complexity or strength..
>
> snip..
>
> >The obvious experiment, then, is to take the device to cryogenic
> >temperatures and see how it performs. If the output still has good
> >statistics, we can suspect that the output does not represent thermal
> >noise at all, but is just a complex digital system. Was such an
> >experiment performed?
>
> No. The circuit depends on many complex factors for randomness, as a
> coin toss does. In some imagined laboratory experiment, it is feasible
> to control all factors, causing non-random results. In commodity
> applications, Large Signal Random Number Generators are sometimes
> superior to small signal based generators and both may appear on a
> single IC.
>
> >>>A signal composed of many oscillators, each doing their own thing, is
> >>>admittedly complex. But complex relationships are not, by themselves,
> >>>cryptographically secure. We could even think to simulate such a
> >>>system numerically, in which case the system is clearly no more than
> >>>yet another pseudorandom state machine waiting to be exposed. And
> >>>while any such simulation might not be exact, it could be close, and
> >>>we could continually adjust the simulation to the reality we do see.
> >>
> >>I hope that you would simulate the 200 oscillator example I gave.
>
> >But your design does not use 200 oscillators, does it?
>
> No it had 3. The 200 oscillator example is for a simplified explanation
> of one source of randomness.
>
> >>Yes,
> >>you can add the +/- 1mV noise and adjust to make it as accurate as you
> >>care to invest time for. I have read your pat statement above several
> >>times recently, and I disagree with it. It is possible to design a
> >>complex circuit that is poorly done and which therefore would fail
> >>tests for randomness.
>
> >Even PSEUDO-random RNG's pass statistical tests. Those tests have
> >nothing to do with cryptographic unpredictability or "strength." Yet
> >strength is what you claim.
>
> Yes it is a strong source, as upcoming product releases are expected to
> show. Just because old PRNGs pass some tests does not mean that new
> designs are bad, as you imply.
>
> >I think you have missed the distinction between unpredictable
> >randomness for cryptography, and ordinary statistical randomness.
>
> A PSRG may be depended upon to produce the same string under certain
> easy to arrange conditions. This RNG does the opposite of that. Two
> sequential random numbers from this circuit would prove that to
> anyone who tests it, most of the time.
>
> Thank you for this polite discussion.

Subject: Re: hardRandNumbGen
Date: Tue, 26 Jan 1999 03:24:11 -1000
From: handWave <real9@complex9.net>
Message-ID: <36ADC1FB.4212@complex9.net>
References: <36ACB68D.34FA34A4@aspi.net>
Newsgroups: sci.crypt
Lines: 116

Trevor Jackson, III wrote:

handWave wrote:
> > I am glad you raised the "handwaves" metaphore, because handwaves are
> > what toss coins. A complex person tosses a coin and you might think it
> > is random. The oscillator RNG in this discussion is directly analogous
> > to a coin toss in many ways. If a coin is not rotating (oscillating)
> > it will fall back into the hand in the same position that it stated from.
> > It is the rotation that helps the randomness, not only the complexity

> > of the nervous system, the elasticity of the skin, and the trembling of
> > the muscles. The rotation should be fast for best results. A juggler
> > could become skilled at non-random coin tosses for one coin that
> > rotates slowly. But if she tosses three coins with rapid rotation than
> > it is likely that random results will occur. If a periodic wind is
> > present in a coin toss, yes, it will influence the outcome, but the
> > result will often be recognizable as a useful random throw, or a throw
> > that was blown away. The same with this RNG.
>
> Human gestures are not a good foundation for system design. There are large,
> industrial concerns that rely upon human-gesture-generated unpredictability.
> Their interest is *not* statistical randomness as we find in simulations,
> games, and Monte Carlo tests (in spite of the latter name). Their interest
> is the same as ours: unpredicability. They are called casinos.

The product I designed was evaluated for casinos by Bally, a potential
customer.

>
> In spite of the fantastic efforts taken to eliminate predictability in games
> of chance human gestures can still dominate the outcomes completely. I'm not
> referring to shuffling cards systematically, but to rolling a roulette ball
> against the wheel so precisely that out of 20 tries a human can obtain a
> predicted outcome (slot 17) 10 times. 50% success. I think that constitutes
> predictability.

Yes, this is like the skilled juggler I described above. The analogy to a
hardRandNumbGen is a skilled hacker who controls the power supply noise,
the clock glitches, the radio beams so that the RNG becomes under his
control. The chip designer must anticipate such antics, and prepare the
module for lunar insertion.

> The hand-eye coordination involved is of an extreme level, requiring decades
> of practice to achieve. But it is real. The complexity of controlling a
> roulette wheel appears to me to be far larger than that of a coin toss. Even
> a fast one.

I dispute this. A coin has one bit of output, a wheel has many bits in
one toss. A wheel is a big target with a smaller bandwidth for RPMs. A
coin has a wider bandwidth, perhaps 1hz to 50 hz, a wheel, from .1 hz to
.5 hz on the initial spin. A coin may be tossed from a rooftop. Wheels
would fracture under such conditions.
>
> Without detailed scrutiny of your design I cannot tell whether it is robust.

I can send you the patent number by private email, upon request posted
here in sci.crypt.

> No inspection of the outcome will convince me it is. However, the design
> philosophy you have expressed leads me to believe there will be weaknesses in
> the system.

Yes there are weaknesses. A moonshot too has weaknesses, and people do
their best to prepare a module for its harsh environment. The payoff is
so sweet, though. It is better to have thrashed and lost some entropy,
than never to have thrashed at all.

>
> > The major source of randomness of this RNG is the unsynchronized
> > nature of multiple oscillators with randomly changing frequencies. This
> > is a large signal phenomenon, which cannot be accurately described
> > mathematically. Similar to a coin toss, many analog variables are
> > involved. These continuous variations of many influences cause seeming
> > randomness. If you can mathematically describe a human coin toss, then
> > so you can with this RNG. But you cannot, and I cannot. That does not
> > invalidate the usefulness of these seed generators, not in this
> > century.
>
> The phrase "randomly changing oscillators" is key to the paragrph above. I
> would like to question the use of the term random in the sense of
> unpredictable. Since the (intermediate) output of the system is driving the
> changes to the oscillators there is a full feedback loop present. This kind
> of system may pass statistical tests for randomness, but it may not be
> unpredictable. The result may "cause seeming randomness", but this is far
> from unpredictability. For instance, how much correlation would you expect
> from a set of such devices initialized identically?

We ran mathematical auto-correlation tests looking exactly for this, and
got good results. This type of multi-oscillator, frequency modulated,
unsynchronized circuit is part analog and part digits. It is susceptible
to realities as a hand exists in realities during a toss. Many subtle
influences come into play, including the capacitance between the moon and
the IC.

>
> Even the presmption that the output would pass statistical tests is
> questionable. One famous gafffe in PRNG design was Knuth's composite
> generator, which he called superrandom. Unfortunately it was a closed loop
> design.

It was a computer program.

>He did not forsee the possibility of cycles so short as to be
> degenerate. All closed loop designs contain this danger. If the hardware
> output is driving the hardware configuration, it is avidly searching for a

> configuration that represents a local minima in its volatility.
>
> Now, given initialization in a configuration near such a local minima, how
> much divergence would we find in the output of a set of these devices?

Good point. This is exactly what we were looking for. The results were excellent. I wave my hands vigorously at this point to emphasize that this type of circuit exists in the real world as we do. It is in the school of hard knocks. It can be defeated. But it has some value as a commodity product in certain well chosen scenarios.

handWave

Subject: Re: hardRandNumbGen
Date: Tue, 26 Jan 1999 13:33:49 +0100
From: Mok-Kong Shen <mok-kong.shen@stud.uni-muenchen.de>
Message-ID: <36ADB62D.E681674F@stud.uni-muenchen.de>
References: <36ADC1FB.4212@complex9.net>
Newsgroups: sci.crypt
Lines: 31

handWave wrote:
>

> > Even the presmption that the output would pass statistical tests is
> > questionable. One famous gafffe in PRNG design was Knuth's composite
> > generator, which he called superrandom. Unfortunately it was a closed loop
> > design.
>
> It was a computer program.

Having previously taken part in discussions in several threads of this group on random number generations, I doubt nevertheless that I have really known an answer to the following question:

If I have two sources of randomness, one software and one hardware, both passing all statistical tests I apply equally well, why should I choose one source in preference to the other? And if additionally I don't know which sequence I get is from software and which is from hardware? (Compare the Turing test.) How does the origin of the sequence affect the workload of the analyst, if the software generation process involves so many parameters that for combinatorical reasons he has no chance of directly dealing with them but has to try to look instead for possible regularities/irregularities in the sequence itself and, by assumption, the sequences from the two sources are of equal statistical quality? (Note that the hardware source is (in my humble opinion) unpredictable simply because there are so many participating 'parameters' that the 'summation' (the end product) becomes unpredictable, cf. the casting of a dice.)

M. K. Shen

Subject: Re: hardRandNumbGen
Date: Tue, 26 Jan 1999 12:28:17 -0500
From: "Trevor Jackson, III" <fullmoon@aspi.net>
Message-ID: <36ADFB30.BB4B07FD@aspi.net>
References: <36ADB62D.E681674F@stud.uni-muenchen.de>
Newsgroups: sci.crypt
Lines: 60

Mok-Kong Shen wrote:

> handWave wrote:
> >
>
> > > Even the presmption that the output would pass statistical tests is
> > > questionable. One famous gafffe in PRNG design was Knuth's composite
> > > generator, which he called superrandom. Unfortunately it was a closed loop
> > > design.
> >
> > It was a computer program.
>
> Having previously taken part in discussions in several threads of
> this group on random number generations, I doubt nevertheless that
> I have really known an answer to the following question:
>
> If I have two sources of randomness, one software and one hardware,
> both passing all statistical tests I apply equally well, why should
> I choose one source in preference to the other? And if additionally
> I don't know which sequence I get is from software and which is from
> hardware? (Compare the Turing test.) How does the origin of the
> sequence affect the workload of the analyst, if the software
> generation process involves so many parameters that for combinatorical
> reasons he has no chance of directly dealing with them but has
> to try to look instead for possible regularities/irregularities in
> the sequence itself and, by assumption, the sequences from the
> two sources are of equal statistical quality? (Note that the
> hardware source is (in my humble opinion) unpredictable simply
> because there are so many participating 'parameters' that the
> 'summation' (the end product) becomes unpredictable, cf. the casting
> of a dice.)

The fundamental reason is that, for security purposes, we have to assume that our opponent can do anything we can do. We can re-run the software and obtain the identical output. We cannot re-run the hardware and get the same output. Thus the hardware is superior.

The deceptive provision in your question is the fact that the sources are hidden. This amounts to security via obscurity. Obscurity fails catastrophicaly when it is breached. A bad thing because the opponent can steal a copy of the software and get every output we will every get. He cannot steal a copy of the machine and get identical outputs to ours.

This line of thought identifies a possible opportunity for Bill Gates; a true marketing genius if there ever was one. Everyone alive in 1980 knew that software was the "plastic" of the decade and that the market for software was going to grow quickly. But no other person alive in 1980 forsaw just how big the market would be for really bad software. Everyone else was concentrating on reasonably good software. This is why Gates is a multi-deca-billionaire.

Now, in crypto, you have identified another case in which people cannot tell whether someone is selling Good Stuff or Really Bad Crap. Since it is not reasonable to distinguish the two, we need an organization to produce a tiny amount of Good Stuff and massive quantities of Really Bad Crap, and sell it all as the former. No one could tell the difference, and, in theory, no one would care.

Bill, are you listening?

Subject: Re: hardRandNumbGen
Date: Tue, 26 Jan 1999 19:25:23 GMT
From: rcktexas@ix.netcom.com (R. Knauer)
Message-ID: <36ae1411.29221428@nntp.ix.netcom.com>
References: <36ADFB30.BB4B07FD@aspi.net>
Newsgroups: sci.crypt
Lines: 45

On Tue, 26 Jan 1999 12:28:17 -0500, "Trevor Jackson, III"
<fullmoon@aspi.net> wrote:

>This line of thought identifies a possible opportunity for Bill Gates; a true
>marketing genius if there ever was one.

I guess you consider Attila the Hun to be a military genius too. :-)

>But no other person alive in 1980 forsaw just how big the market would be
>for really bad software.

Hell, the auto industry knew that way before Gates used it in the S/W
industry. He just took the same marketing concepts used by Henry Ford
and built the same kind of fortune.

"You can have any color Model T you want as long as it runs on
Windows."

>Now, in crypto, you have identified another case in which people cannot tell
>whether someone is selling Good Stuff or Really Bad Crap. Since it is not
>reasonable to distinguish the two, we need an organization to produce a tiny amount
>of Good Stuff and massive quantities of Really Bad Crap, and sell it all as the
>former. No one could tell the difference, and, in theory, no one would care.

Soon Gates is gonna retire all his programmers at MicroShaft and
install a TRNG to produce code. And now that his beta test force is
big enough, he can partition the outputs and see what runs
experimentally.

Depending on which beta test group(s) order the next "revision", he
can decide what to put in shrinkwrap. If it gets to the Windows Logo,
it is good enough for the consuming public.

If they don't like it, let them run UNIX.

>Bill, are you listening?

HA! Unka Bill is too busy working on his new TRNG.

Bob Knauer

"An honest man can feel no pleasure in the exercise of power over
his fellow citizens."
--Thomas Jefferson

Subject: Re: hardRandNumbGen
Date: Wed, 27 Jan 1999 15:30:05 +0100
From: Mok-Kong Shen <mok-kong.shen@stud.uni-muenchen.de>
Message-ID: <36AF22ED.9D0E25B0@stud.uni-muenchen.de>
References: <36ADFB30.BB4B07FD@aspi.net>
Newsgroups: sci.crypt
Lines: 21

Trevor Jackson, III wrote:
>

> The fundamental reason is that, for security purposes, we have to assume that our
> opponent can do anything we can do. We can re-run the software and obtain the
> identical output. We cannot re-run the hardware and get the same output. Thus the
> hardware is superior.
>
> The deceptive provision in your question is the fact that the sources are hidden.
> This amounts to security via obscurity. Obscurity fails catastrophicaly when it is
> breached. A bad thing because the opponent can steal a copy of the software and
> get every output we will every get. He cannot steal a copy of the machine and get
> identical outputs to ours.

If you produce some sequence with a sufficiently good algorithm with
a sufficiently long key and later forget that key, even you wouldn't
be able to reproduce the sequence. As to stealing I suppose it is
irrelevant in the present context. (If you have a one-time pad and
that got stolen (copied), then what?)

M. K. Shen

Subject: Re: hardRandNumbGen
Date: Tue, 26 Jan 1999 17:56:39 GMT
From: rcktexas@ix.netcom.com (R. Knauer)
Message-ID: <36ae019b.24495482@nntp.ix.netcom.com>
References: <36ADB62D.E681674F@stud.uni-muenchen.de>
Newsgroups: sci.crypt
Lines: 16

On Tue, 26 Jan 1999 13:33:49 +0100, Mok-Kong Shen
<mok-kong.shen@stud.uni-muenchen.de> wrote:

>If I have two sources of randomness, one software and one hardware,
>both passing all statistical tests I apply equally well, why should
>I choose one source in preference to the other?

Why do you persist in believing that statistical tests have anything
to do with randomness in cryptography?

Bob Knauer

"An honest man can feel no pleasure in the exercise of power over
his fellow citizens."
--Thomas Jefferson

Subject: Re: hardRandNumbGen
Date: Tue, 26 Jan 1999 19:24:18 +0100
From: Mok-Kong Shen <mok-kong.shen@stud.uni-muenchen.de>
Message-ID: <36AE0852.16B9A95F@stud.uni-muenchen.de>
References: <36ae019b.24495482@nntp.ix.netcom.com>
Newsgroups: sci.crypt
Lines: 18

R. Knauer wrote:
>
> On Tue, 26 Jan 1999 13:33:49 +0100, Mok-Kong Shen
> <mok-kong.shen@stud.uni-muenchen.de> wrote:
>
> >If I have two sources of randomness, one software and one hardware,
> >both passing all statistical tests I apply equally well, why should
> >I choose one source in preference to the other?
>
> Why do you persist in believing that statistical tests have anything
> to do with randomness in cryptography?

Tell me what other (better) tools are available for me to make
the decision. These are simply easy to obtain, as far as my
humble knowledge goes. Please kindly give your recipe to cope with
the situation I described. Thanks in advance.

M. K. Shen

Subject: Re: hardRandNumbGen
Date: Tue, 26 Jan 1999 19:33:24 GMT
From: rcktexas@ix.netcom.com (R. Knauer)
Message-ID: <36ae16a8.29884341@nntp.ix.netcom.com>
References: <36AE0852.16B9A95F@stud.uni-muenchen.de>
Newsgroups: sci.crypt
Lines: 41

On Tue, 26 Jan 1999 19:24:18 +0100, Mok-Kong Shen
<mok-kong.shen@stud.uni-muenchen.de> wrote:

>> Why do you persist in believing that statistical tests have anything
>> to do with randomness in cryptography?

>Tell me what other (better) tools are available for me to make
>the decision.

If I told you that there are none, would you believe me?

>These are simply easy to obtain, as far as my
>humble knowledge goes.

So is snake oil.

>Please kindly give your recipe to cope with
>the situation I described. Thanks in advance.

Learn what crypto-grade randomness is. The concept is deceptively simple once you understand it. But first you have to give up all other definitions of randomness from other fields like statistics.

The key to understanding is that randomness depends on the generation process, not the numbers themselves. The number 000...0 fails all sorts of statistical tests, but can be a random number if it is generated by a TRNG. Until you analyze the method of generation, you cannot know.

A TRNG is a physical device that is capable of generating all possible sequences of a given finite length equiprobably. If you understand that, then you will understand crypto-grade randomness - and, as another poster pointed out yesterday, you will also understand cryptography.

Bob Knauer

"An honest man can feel no pleasure in the exercise of power over his fellow citizens."
--Thomas Jefferson

Subject: Re: hardRandNumbGen
Date: Wed, 27 Jan 1999 15:38:29 +0100
From: Mok-Kong Shen <mok-kong.shen@stud.uni-muenchen.de>
Message-ID: <36AF24E5.95D5D7F9@stud.uni-muenchen.de>
References: <36ae16a8.29884341@nntp.ix.netcom.com>
Newsgroups: sci.crypt
Lines: 17

R. Knauer wrote:

> A TRNG is a physical device that is capable of generating all possible
> sequences of a given finite length equiprobably. If you understand
> that, then you will understand crypto-grade randomness - and, as
> another poster pointed out yesterday, you will also understand
> cryptography.

Excellent! Then tell me HOW to get such a physical device that PROVABLY is capable of generating all possible sequences of a given finite length equiprobalbly.

Secondly, your equiprobability is not at all sufficient. If the said given finite length is 2, is a physical divice outputting 0001101100011011..... a TRNG?????

M. K. Shen

Subject: Re: hardRandNumbGen
Date: 27 Jan 1999 09:51:46 -0500
From: juola@mathcs.duq.edu (Patrick Juola)
Message-ID: <78n962lii1@quine.mathcs.duq.edu>
References: <36AF24E5.95D5D7F9@stud.uni-muenchen.de>
Newsgroups: sci.crypt
Lines: 40

In article <36AF24E5.95D5D7F9@stud.uni-muenchen.de>,
Mok-Kong Shen <mok-kong.shen@stud.uni-muenchen.de> wrote:
>R. Knauer wrote:
>
>> A TRNG is a physical device that is capable of generating all possible
>> sequences of a given finite length equiprobably. If you understand
>> that, then you will understand crypto-grade randomness - and, as
>> another poster pointed out yesterday, you will also understand
>> cryptography.
>
>Excellent! Then tell me HOW to get such a physical device that
>PROVABLY is capable of generating all possible sequences of a given
>finite length equiprobalbly.

You can't. Tell me how you can build a plane that will *provably* fly equally stably in any direction.

>Secondly, your equiprobability is not at all sufficient. If
>the said given finite length is 2, is a physical divice outputting
>0001101100011011..... a TRNG?????

You can't tell. You've framed the question such that it's unanswerable due to insufficient information :

There is a coffee cup on the southeast corner of my desk. If
it is approximately 1/3 full, what is written on the outside
of the cup?

What kind of mileage does a blue car get?

However, the fact that you've asked a dumb question doesn't mean that
the concepts aren't useful -- both paint color and mileage are
important in describing and evaluating cars. But they're not connected
the way you think they are.

The fact that you're repeatedly asking the same dumb question does,
however, suggest that you're not really interested in the answer.

-kitten

Subject: Re: hardRandNumbGen
Date: Wed, 27 Jan 1999 16:12:44 +0100
From: Mok-Kong Shen <mok-kong.shen@stud.uni-muenchen.de>
Message-ID: <36AF2CEC.B5684684@stud.uni-muenchen.de>
References: <78n962lii1@quine.mathcs.duq.edu>
Newsgroups: sci.crypt
Lines: 11

Patrick Juola wrote:
>

> The fact that you're repeatedly asking the same dumb question does,
> however, suggest that you're not really interested in the answer.

The origninal purpose is evidently: Since there can't be an good
answer, one can't claim hardware sequences are always to be preferred
to software sequences.

M. K. Shen

Subject: Re: hardRandNumbGen
Date: 27 Jan 1999 10:30:16 -0500
From: juola@mathcs.duq.edu (Patrick Juola)
Message-ID: <78nbe8llm1@quine.mathcs.duq.edu>
References: <36AF2CEC.B5684684@stud.uni-muenchen.de>
Newsgroups: sci.crypt
Lines: 27

In article <36AF2CEC.B5684684@stud.uni-muenchen.de>,
Mok-Kong Shen <mok-kong.shen@stud.uni-muenchen.de> wrote:
>Patrick Juola wrote:
>>
>
>> The fact that you're repeatedly asking the same dumb question does,
>> however, suggest that you're not really interested in the answer.
>
>The origninal purpose is evidently: Since there can't be an good
>answer, one can't claim hardware sequences are always to be preferred
>to software sequences.

Your claim above is untrue. I can prove that there can't be a good
s/w sequence running on a deterministic machine. But I can't do that
merely by inspecting any finite sample of outputs -- I have to look
at the generators to do it. Of course, any bad PRNG can be implemented
either in h/w or s/w, so just because something is in h/w doesn't make
it good.

More accurately : one can't claim hardware sequences are always to
be preferred to software sequences *on the basis of a statistical
analysis of a finite set of output sequences.*

But this is unsurprising. I can't tell you the gas mileage by looking
at the color of the paint, either.

-kitten

Subject: Re: hardRandNumbGen
Date: Wed, 27 Jan 1999 16:48:46 +0100
From: Mok-Kong Shen <mok-kong.shen@stud.uni-muenchen.de>
Message-ID: <36AF355E.DA6A3DBC@stud.uni-muenchen.de>
References: <78nbe8llm1@quine.mathcs.duq.edu>
Newsgroups: sci.crypt
Lines: 11

Patrick Juola wrote:
>

> More accurately : one can't claim hardware sequences are always to
> be preferred to software sequences *on the basis of a statistical
> analysis of a finite set of output sequences.*

The issue is: Are there other sound scientific basis to claim the
said preference.

M. K. Shen

Subject: Re: hardRandNumbGen
Date: 27 Jan 1999 12:04:02 -0500
From: juola@mathcs.duq.edu (Patrick Juola)
Message-ID: <78ngu3lsg1@quine.mathcs.duq.edu>
References: <36AF355E.DA6A3DBC@stud.uni-muenchen.de>
Newsgroups: sci.crypt
Lines: 16

In article <36AF355E.DA6A3DBC@stud.uni-muenchen.de>,
Mok-Kong Shen <mok-kong.shen@stud.uni-muenchen.de> wrote:
>Patrick Juola wrote:
>>
>
>> More accurately : one can't claim hardware sequences are always to
>> be preferred to software sequences *on the basis of a statistical
>> analysis of a finite set of output sequences.*
>
>The issue is: Are there other sound scientific basis to claim the
>said preference.

And the answer is : yes, if your goal is to provide unbounded
degrees of security for messages of unbounded length.

-kitten

Subject: Re: hardRandNumbGen
Date: Wed, 27 Jan 1999 18:20:22 +0100
From: Mok-Kong Shen <mok-kong.shen@stud.uni-muenchen.de>
Message-ID: <36AF4AD6.1EC74EF0@stud.uni-muenchen.de>
References: <78ngu3lsg1@quine.mathcs.duq.edu>
Newsgroups: sci.crypt
Lines: 27

Patrick Juola wrote:
>
> In article <36AF355E.DA6A3DBC@stud.uni-muenchen.de>,
> Mok-Kong Shen <mok-kong.shen@stud.uni-muenchen.de> wrote:
> >Patrick Juola wrote:
> >>
> >
> >> More accurately : one can't claim hardware sequences are always to
> >> be preferred to software sequences *on the basis of a statistical
> >> analysis of a finite set of output sequences.*
> >
> >The issue is: Are there other sound scientific basis to claim the
> >said preference.
>
> And the answer is : yes, if your goal is to provide unbounded
> degrees of security for messages of unbounded length.

I would be happy with a weaker goal, i.e. for messages of a
certain finite length. Could you provide the requested sound
scientific basis? Note that I am going to use the sequences
in practical applications. So any claimed degree of security
has be shown with a practical algorithm. I am also prepared to
weaken the goal further to 'bounded degree of security' if you
can give a precise definition of 'degree of security' that is
satisfactory for the practice.

M. K. Shen

Subject: Re: hardRandNumbGen
Date: 27 Jan 1999 14:27:50 -0500
From: juola@mathcs.duq.edu (Patrick Juola)
Message-ID: <78npbm$m42$1@quine.mathcs.duq.edu>
References: <36AF4AD6.1EC74EF0@stud.uni-muenchen.de>
Newsgroups: sci.crypt
Lines: 67

In article <36AF4AD6.1EC74EF0@stud.uni-muenchen.de>,
Mok-Kong Shen <mok-kong.shen@stud.uni-muenchen.de> wrote:
>Patrick Juola wrote:
>>
>> In article <36AF355E.DA6A3DBC@stud.uni-muenchen.de>,
>> Mok-Kong Shen <mok-kong.shen@stud.uni-muenchen.de> wrote:
>> >Patrick Juola wrote:
>> >>
>> >
>> >> More accurately : one can't claim hardware sequences are always to
>> >> be preferred to software sequences *on the basis of a statistical
>> >> analysis of a finite set of output sequences.*
>> >
>> >The issue is: Are there other sound scientific basis to claim the
>> >said preference.
>>
>> And the answer is : yes, if your goal is to provide unbounded
>> degrees of security for messages of unbounded length.

>
>I would be happy with a weaker goal, i.e. for messages of a
>certain finite length. Could you provide the requested sound
>scientific basis?

Sure. If the key is "as random as" the message, then Shannon's proof of secrecy goes through. In particular, if your messages are bounded by a given length N, then if you can get N bits of randomness, from whatever source, hardware or software, then you can achieve perfect secrecy.

How you get them is, of course, your problem. The difficulty is in *proving* that a given sequence of N bits is contains N bits of randomness (or more formally that a given generator produces exactly random bits). But it's fairly easy to gather *MORE* than N bits -- as much more as you feel confident that it is unlikely to be more less than N bits of randomness in the resulting sample.

Furthermore, I note that "sound scientific basis" doesn't necessarily rely on a formal, mathematical proof. We use the acceleration of gravity g = 9.8 m/sec on the basis of experiment rather than any first-principle analysis. Similar experiments show that, for instance, English text contains just over one bit of randomness per character. If you need a thousand bits of randomness, get a thousand characters of English text from a secure source, distill them appropriately with a trusted hashing function. Better yet, get 1500 characters to allow for sloppy engineering -- you'd never run a wire at its rated wattage, would you? Take the resulting 1000 bit string, XOR it with the plaintext, and voila. A scientifically sound method of securing 1000 bit secrets.

>I am also prepared to
>weaken the goal further to 'bounded degree of security' if you
>can give a precise definition of 'degree of security' that is
>satisfactory for the practice.

Well, the usual definition is "work factor" -- the ratio of work necessary to read a message without the key vs. with the key. Again, "sound scientific basis" does not necessarily rely on proof; if you are willing to accept (as many scientists do) that RSA is as secure as factoring, then the work factor to crack an RSA code can be made as large as you like by raising the modulus appropriately. If you don't believe that RSA is as secure as factoring.... well, there are other methods out there with various conjectures about their difficulty of solution. If you don't believe *any* conjectures, you're arguably in the same camp as people who don't really believe that the force of gravity is constant just because it's always been constant so far....

-kitten

Subject: Re: hardRandNumbGen
Date: Wed, 27 Jan 1999 23:54:24 GMT
From: rcktexas@ix.netcom.com (R. Knauer)
Message-ID: <36afa6cd.50106439@nntp.ix.netcom.com>
References: <78npbm$m42$1@quine.mathcs.duq.edu>
Newsgroups: sci.crypt
Lines: 21

On 27 Jan 1999 14:27:50 -0500, juola@mathcs.duq.edu (Patrick Juola)
wrote:

>English text contains just over one bit of randomness per character.
>If you need a thousand bits of randomness, get a thousand characters
>of English text from a secure source, distill them appropriately with
>a trusted hashing function. Better yet, get 1500 characters to allow
>for sloppy engineering -- you'd never run a wire at its rated wattage,
>would you? Take the resulting 1000 bit string, XOR it with the plaintext,
>and voila. A scientifically sound method of securing 1000 bit secrets.

You once said that such a system was vulnerable to a Bayesian attack.
Have you changed your mind?

Bob Knauer

"No Freeman shall ever be debarred the use of arms. The strongest
reason for the people to retain the right to keep and bear arms is,
as a last resort, to protect themselves against tyranny in government."
--Thomas Jefferson

Subject: Re: hardRandNumbGen
Date: 28 Jan 1999 11:25:33 -0500
From: juola@mathcs.duq.edu (Patrick Juola)
Message-ID: <78q31tngq1@quine.mathcs.duq.edu>
References: <36afa6cd.50106439@nntp.ix.netcom.com>
Newsgroups: sci.crypt
Lines: 25

In article <36afa6cd.50106439@nntp.ix.netcom.com>,
R. Knauer <rcktexas@ix.netcom.com> wrote:
>On 27 Jan 1999 14:27:50 -0500, juola@mathcs.duq.edu (Patrick Juola)
>wrote:

>
>>English text contains just over one bit of randomness per character.
>>If you need a thousand bits of randomness, get a thousand characters
>>of English text from a secure source, distill them appropriately with
>>a trusted hashing function. Better yet, get 1500 characters to allow
>>for sloppy engineering -- you'd never run a wire at its rated wattage,
>>would you? Take the resulting 1000 bit string, XOR it with the plaintext,
>>and voila. A scientifically sound method of securing 1000 bit secrets.
>
>You once said that such a system was vulnerable to a Bayesian attack.
>Have you changed your mind?

No. The key point here is that the key is as large as -- larger than,
in fact -- the plaintext. Such a system *would* be vulnerable if you
were using it to secure secrets larger than 1000 bits. But as long
as the plaintext is finite *AND BOUNDED*, if you can get key material
to exceed that bound, you can get perfect secrecy.

But few of us have bounded secrets.

-kitten

Subject: Re: hardRandNumbGen
Date: Thu, 28 Jan 1999 23:40:37 GMT
From: rcktexas@ix.netcom.com (R. Knauer)
Message-ID: <36b0f4c7.11366854@nntp.ix.netcom.com>
References: <78q31tngq1@quine.mathcs.duq.edu>
Newsgroups: sci.crypt
Lines: 22

On 28 Jan 1999 11:25:33 -0500, juola@mathcs.duq.edu (Patrick Juola)
wrote:

>as long
>as the plaintext is finite *AND BOUNDED*, if you can get key material
>to exceed that bound, you can get perfect secrecy.

>But few of us have bounded secrets.

You are being uncharacteristically obscure.

Please elaborate on the concepts of "bounded", "unbounded" and how
they apply to a "bounded secret". And just how is a plaintext
"bounded", given that it is finite in length?

Bob Knauer

"No Freeman shall ever be debarred the use of arms. The strongest
reason for the people to retain the right to keep and bear arms is,
as a last resort, to protect themselves against tyranny in government."
--Thomas Jefferson

Subject: Re: hardRandNumbGen
Date: 29 Jan 1999 08:56:25 -0500
From: juola@mathcs.duq.edu (Patrick Juola)
Message-ID: <78sem9ori1@quine.mathcs.duq.edu>
References: <36b0f4c7.11366854@nntp.ix.netcom.com>
Newsgroups: sci.crypt
Lines: 37

In article <36b0f4c7.11366854@nntp.ix.netcom.com>,
R. Knauer <rcktexas@ix.netcom.com> wrote:
>On 28 Jan 1999 11:25:33 -0500, juola@mathcs.duq.edu (Patrick Juola)
>wrote:
>
>>as long
>>as the plaintext is finite *AND BOUNDED*, if you can get key material
>>to exceed that bound, you can get perfect secrecy.
>
>>But few of us have bounded secrets.
>
>You are being uncharacteristically obscure.
>
>Please elaborate on the concepts of "bounded", "unbounded" and how
>they apply to a "bounded secret". And just how is a plaintext
>"bounded", given that it is finite in length?

The idea behind a bounded plaintext is fairly simple. Just say
to yourself that you will never, EVER, in your entire life,
encrypt a document larger than X with a single key. Splitting
a big document into two into order to make it smaller doesn't
count, as you need two different keys in that case.

X is, then, "the bound." And it's a measure of how much work you
need to generate the key for each and every message you send --
so make it low.

The difference between bounded and finite is simple -- with finite,
plaintexts, I know that my articles will eventually end, but I don't
know when beforehand. With a bounded plantext, I set myself a rule
beforehand that I won't go over 30 lines, or 300, or 3 million, whatever

and stick to that rule.

Can you promise yourself that you'll never want to Email yourself a copy of Microsoft Word?

-kitten

Subject: Re: hardRandNumbGen
Date: Thu, 28 Jan 1999 16:10:36 +0100
From: Mok-Kong Shen <mok-kong.shen@stud.uni-muenchen.de>
Message-ID: <36B07DEC.7D4DE9EE@stud.uni-muenchen.de>
References: <78npbm$m42$1@quine.mathcs.duq.edu>
Newsgroups: sci.crypt
Lines: 59

Patrick Juola wrote:

> Sure. If the key is "as random as" the message, then Shannon's
> proof of secrecy goes through. In particular, if your messages
> are bounded by a given length N, then if you can get N bits of
> randomness, from whatever source, hardware or software, then
> you can achieve perfect secrecy.
>
> How you get them is, of course, your problem. The difficulty is
> in *proving* that a given sequence of N bits is contains N bits
> of randomness (or more formally that a given generator produces
> exactly random bits). But it's fairly easy to gather *MORE* than
> N bits -- as much more as you feel confident that it is unlikely
> to be more less than N bits of randomness in the resulting sample.
>
> Furthermore, I note that "sound scientific basis" doesn't necessarily
> rely on a formal, mathematical proof. We use the acceleration of
> gravity g = 9.8 m/sec on the basis of experiment rather than any
> first-principle analysis. Similar experiments show that, for instance,
> English text contains just over one bit of randomness per character.
> If you need a thousand bits of randomness, get a thousand characters
> of English text from a secure source, distill them appropriately with
> a trusted hashing function. Better yet, get 1500 characters to allow
> for sloppy engineering -- you'd never run a wire at its rated wattage,
> would you? Take the resulting 1000 bit string, XOR it with the plaintext,
> and voila. A scientifically sound method of securing 1000 bit secrets.

Thank you for the very lucid description of a sound standpoint in practice ('applied' cryptography as against 'theoretical' cryptography). We must be realistic, since theoretical stuffs may not be realizable in the real world and since 'absulute' security is never required (does it matter if a cipher is cracked after 100 years?) Incidentally, in another thread I also suggested distiling bit sequences out of natural language texts as raw materials.

>
> Well, the usual definition is "work factor" -- the ratio of work
> necessary to read a message without the key vs. with the key.
> Again, "sound scientific basis" does not necessarily rely on proof;
> if you are willing to accept (as many scientists do) that RSA is
> as secure as factoring, then the work factor to crack an RSA code
> can be made as large as you like by raising the modulus appropriately.
> If you don't believe that RSA is as secure as factoring.... well, there
> are other methods out there with various conjectures about their
> difficulty of solution. If you don't believe *any* conjectures, you're
> arguably in the same camp as people who don't really believe that
> the force of gravity is constant just because it's always been constant
> so far....

I appreciate your opinions and in particular agree with you that formal proofs are not always needed but can be substituted with 'practical' yet scientifically sound procedures. One should never be pedantic but one certainly should not be on the other hand a 'believer' of 'religious assertions' (totally unfounded big words of someone).

M. K. Shen

M. K. Shen

Subject: Re: hardRandNumbGen
Date: Wed, 27 Jan 1999 16:17:41 GMT
From: rcktexas@ix.netcom.com (R. Knauer)
Message-ID: <36af3bd0.22717926@nntp.ix.netcom.com>
References: <78nbe8llm1@quine.mathcs.duq.edu>
Newsgroups: sci.crypt
Lines: 17

On 27 Jan 1999 10:30:16 -0500, juola@mathcs.duq.edu (Patrick Juola) wrote:

>But this is unsurprising. I can't tell you the gas mileage by looking
>at the color of the paint, either.

There were certain colors that were used exclusively on the Volkswagen Beetle. That would have given you a strong enough clue to infer the gas mileage, assuming standard operating conditions.

Bob Knauer

"No Freeman shall ever be debarred the use of arms. The strongest
reason for the people to retain the right to keep and bear arms is,
as a last resort, to protect themselves against tyranny in government."
--Thomas Jefferson

Subject: Re: hardRandNumbGen
Date: Wed, 27 Jan 1999 15:43:43 GMT
From: rcktexas@ix.netcom.com (R. Knauer)
Message-ID: <36af340c.20729677@nntp.ix.netcom.com>
References: <36AF2CEC.B5684684@stud.uni-muenchen.de>
Newsgroups: sci.crypt
Lines: 19

On Wed, 27 Jan 1999 16:12:44 +0100, Mok-Kong Shen
<mok-kong.shen@stud.uni-muenchen.de> wrote:

>> The fact that you're repeatedly asking the same dumb question does,
>> however, suggest that you're not really interested in the answer.

>The origninal purpose is evidently: Since there can't be an good
>answer, one can't claim hardware sequences are always to be preferred
>to software sequences.

See! What did I tell you.

Bob Knauer

"No Freeman shall ever be debarred the use of arms. The strongest
reason for the people to retain the right to keep and bear arms is,
as a last resort, to protect themselves against tyranny in government."
--Thomas Jefferson

Subject: Re: hardRandNumbGen
Date: Wed, 27 Jan 1999 15:40:53 GMT
From: rcktexas@ix.netcom.com (R. Knauer)
Message-ID: <36af31b5.20130325@nntp.ix.netcom.com>
References: <78n962lii1@quine.mathcs.duq.edu>
Newsgroups: sci.crypt
Lines: 36

On 27 Jan 1999 09:51:46 -0500, juola@mathcs.duq.edu (Patrick Juola)
wrote:

>In article <36AF24E5.95D5D7F9@stud.uni-muenchen.de>,
>Mok-Kong Shen <mok-kong.shen@stud.uni-muenchen.de> wrote:

>>Secondly, your equiprobability is not at all sufficient. If
>>the said given finite length is 2, is a physical divice outputting
>>0001101100011011..... a TRNG?????

>The fact that you're repeatedly asking the same dumb question does,
>however, suggest that you're not really interested in the answer.

The answer that the poster wants to hear is: Because TRNGs are not
Perfect, PRNGs are just as good.

What he fails to appreciate is that there is a fundamental difference
between a TRNG and a PRNG. That is because he fails to realize that a
crypto-grade random number is characterized by the generation process,
not the number itself.

IOW, according to the poster, regardless of whether a number is
generated by a TRNG or a PRNG, if it passes some statistical tests
(that only work on infinite numbers), then it makes no difference what
the method of generation is.

Maybe there needs to be a law that a student must take cryptography
before statistics. :-)

Bob Knauer

"No Freeman shall ever be debarred the use of arms. The strongest
reason for the people to retain the right to keep and bear arms is,
as a last resort, to protect themselves against tyranny in government."
--Thomas Jefferson

Subject: Re: hardRandNumbGen
Date: Wed, 27 Jan 1999 16:52:30 +0100
From: Mok-Kong Shen <mok-kong.shen@stud.uni-muenchen.de>
Message-ID: <36AF363E.45670214@stud.uni-muenchen.de>
References: <36af31b5.20130325@nntp.ix.netcom.com>
Newsgroups: sci.crypt
Lines: 12

R. Knauer wrote:
>

> What he fails to appreciate is that there is a fundamental difference
> between a TRNG and a PRNG. That is because he fails to realize that a
> crypto-grade random number is characterized by the generation process,
> not the number itself.

Where is the proof of 'if the generation process is hardware then
it is crypto-grade, otherwise it is not'??

M. K. Shen

Subject: Re: hardRandNumbGen
Date: Wed, 27 Jan 1999 16:45:09 GMT
From: rcktexas@ix.netcom.com (R. Knauer)
Message-ID: <36af3f70.23645400@nntp.ix.netcom.com>
References: <36AF363E.45670214@stud.uni-muenchen.de>
Newsgroups: sci.crypt
Lines: 41

On Wed, 27 Jan 1999 16:52:30 +0100, Mok-Kong Shen
<mok-kong.shen@stud.uni-muenchen.de> wrote:

>Where is the proof of 'if the generation process is hardware then
>it is crypto-grade, otherwise it is not'??

There is no proof of the first part, since PRNGs can be implemented in
H/W, like shift register PRNGs.

The proof of the second part comes from an analysis of what makes a
PRNG behave the way it does. It is based on an algorithm, which means
that its output is deterministic, and that means that there could be
vulnerability in using it. For example, if it did not output all
possible sequences of a given finite length equiprobably but started
repeating the sequneces, then it fails the definition for a TRNG.

If a PRNG only puts out a few sequences most of the time, then it is
obviously worthless. That takes care of the equiprobable part of the
TRNG specification.

Assuming that the outputs of the PRNG are equiprobable, if the PRNG is
seeded with a number of length K that is smaller than the output
needed to encrypt the message of length N, then it can only generate
as many sequences as the seed will allow, which is not the same as all
possible sequences.

If K<N, then only 2^K possible plaintexts are contained in the
ciphertext, instead of 2^N. That makes the cryptanalyst's job a lot
easier, especially when the message is longer than the unicity
distance. In that case, there is only 1 plaintext that is
intelligible. When the cryptanalyst finds it, he knows with certainty
that it is the intended message. That is not the case when using a pad
that is generated from a TRNG, where the full 2^N outputs possible.

Bob Knauer

"No Freeman shall ever be debarred the use of arms. The strongest
reason for the people to retain the right to keep and bear arms is,
as a last resort, to protect themselves against tyranny in government."
--Thomas Jefferson

Subject: Re: hardRandNumbGen
Date: Wed, 27 Jan 1999 18:01:00 +0100
From: Mok-Kong Shen <mok-kong.shen@stud.uni-muenchen.de>
Message-ID: <36AF464C.AD52C35D@stud.uni-muenchen.de>
References: <36af3f70.23645400@nntp.ix.netcom.com>
Newsgroups: sci.crypt
Lines: 41

R. Knauer wrote:
>
> On Wed, 27 Jan 1999 16:52:30 +0100, Mok-Kong Shen
> <mok-kong.shen@stud.uni-muenchen.de> wrote:
>
> >Where is the proof of 'if the generation process is hardware then
> >it is crypto-grade, otherwise it is not'??
>
> There is no proof of the first part, since PRNGs can be implemented in
> H/W, like shift register PRNGs.
>
> The proof of the second part comes from an analysis of what makes a
> PRNG behave the way it does. It is based on an algorithm, which means
> that its output is deterministic, and that means that there could be
> vulnerability in using it. For example, if it did not output all
> possible sequences of a given finite length equiprobably but started
> repeating the sequneces, then it fails the definition for a TRNG.
>
> If a PRNG only puts out a few sequences most of the time, then it is
> obviously worthless. That takes care of the equiprobable part of the
> TRNG specification.

>
> Assuming that the outputs of the PRNG are equiprobable, if the PRNG is
> seeded with a number of length K that is smaller than the output
> needed to encrypt the message of length N, then it can only generate
> as many sequences as the seed will allow, which is not the same as all
> possible sequences.
>
> If K<N, then only 2^K possible plaintexts are contained in the
> ciphertext, instead of 2^N. That makes the cryptanalyst's job a lot
> easier, especially when the message is longer than the unicity
> distance. In that case, there is only 1 plaintext that is
> intelligible. When the cryptanalyst finds it, he knows with certainty
> that it is the intended message. That is not the case when using a pad
> that is generated from a TRNG, where the full 2^N outputs possible.

Please note I don't claim PRNGs are good. I simply doubt that
hardward generators are good because I have no tools to determine
that they are good, except by using statistical tools.

M. K. Shen

Subject: Re: hardRandNumbGen
Date: Wed, 27 Jan 1999 18:02:02 GMT
From: rcktexas@ix.netcom.com (R. Knauer)
Message-ID: <36af546f.29020769@nntp.ix.netcom.com>
References: <36AF464C.AD52C35D@stud.uni-muenchen.de>
Newsgroups: sci.crypt
Lines: 18

On Wed, 27 Jan 1999 18:01:00 +0100, Mok-Kong Shen
<mok-kong.shen@stud.uni-muenchen.de> wrote:

>Please note I don't claim PRNGs are good. I simply doubt that
>hardward generators are good because I have no tools to determine
>that they are good, except by using statistical tools.

You must be skilled at designing a TRNG. Statistical tools are
worthless.

Bob Knauer

"No Freeman shall ever be debarred the use of arms. The strongest
reason for the people to retain the right to keep and bear arms is,
as a last resort, to protect themselves against tyranny in government."
--Thomas Jefferson

Subject: Re: hardRandNumbGen
Date: Wed, 27 Jan 1999 19:19:57 +0100
From: Mok-Kong Shen <mok-kong.shen@stud.uni-muenchen.de>
Message-ID: <36AF58CD.C158C845@stud.uni-muenchen.de>
References: <36af546f.29020769@nntp.ix.netcom.com>
Newsgroups: sci.crypt
Lines: 20

R. Knauer wrote:
>
> On Wed, 27 Jan 1999 18:01:00 +0100, Mok-Kong Shen
> <mok-kong.shen@stud.uni-muenchen.de> wrote:
>
> >Please note I don't claim PRNGs are good. I simply doubt that
> >hardward generators are good because I have no tools to determine
> >that they are good, except by using statistical tools.
>
> You must be skilled at designing a TRNG. Statistical tools are
> worthless.

How do you measure or test the skill of a person designing a TRNG?
Using some tests of the psychologists?? More precisely, how do
you show through a test of skill that the resulting TRNG designed
by that person has a certain 'crypto-grade', even if you could define
that 'crypto-grade' scientifically precisely which I guess you can't.
Are you excluding that a skilled person could ever make mistakes??

M. K. Shen

Subject: Re: hardRandNumbGen
Date: 27 Jan 1999 14:29:25 -0500
From: juola@mathcs.duq.edu (Patrick Juola)
Message-ID: <78npel$m4j$1@quine.mathcs.duq.edu>
References: <36AF58CD.C158C845@stud.uni-muenchen.de>
Newsgroups: sci.crypt
Lines: 21

In article <36AF58CD.C158C845@stud.uni-muenchen.de>,
Mok-Kong Shen <mok-kong.shen@stud.uni-muenchen.de> wrote:
>R. Knauer wrote:
>>
>> On Wed, 27 Jan 1999 18:01:00 +0100, Mok-Kong Shen

>> <mok-kong.shen@stud.uni-muenchen.de> wrote:
>>
>> >Please note I don't claim PRNGs are good. I simply doubt that
>> >hardward generators are good because I have no tools to determine
>> >that they are good, except by using statistical tools.
>>
>> You must be skilled at designing a TRNG. Statistical tools are
>> worthless.
>
>How do you measure or test the skill of a person designing a TRNG?

The same way you test the skill of the architect who designed your
building.

-kitten

Subject: Re: hardRandNumbGen
Date: Wed, 27 Jan 1999 23:26:06 GMT
From: rcktexas@ix.netcom.com (R. Knauer)
Message-ID: <36af9b9f.47244373@nntp.ix.netcom.com>
References: <36AF58CD.C158C845@stud.uni-muenchen.de>
Newsgroups: sci.crypt
Lines: 37

On Wed, 27 Jan 1999 19:19:57 +0100, Mok-Kong Shen
<mok-kong.shen@stud.uni-muenchen.de> wrote:

>How do you measure or test the skill of a person designing a TRNG?
>Using some tests of the psychologists??

Make him pee in a bottle and if he does, fire him. If he throws the
bottle at you, hire him.

Skilled people fiercely guard their personal rights as humans. Only
sheep seek the comfort of conformity - and you do not want some sheep
designing a TRNG for you.

>More precisely, how do
>you show through a test of skill that the resulting TRNG designed
>by that person has a certain 'crypto-grade', even if you could define
>that 'crypto-grade' scientifically precisely which I guess you can't.
>Are you excluding that a skilled person could ever make mistakes??

Get several skilled people to check the design and the actual TRNG.
That's why there are standards committees.

But if you attempt to rely on statistical tests on the output alone,
you are not going to know much of anything - except in the trivial
case of a shorted or floating output. But that's it.

If you could determine the randomness of a finite sequence
algorithmically, you could solve both Godel's incompleteness problem
and Turing's halting problem. See Chaitin for the reasons why.

Bob Knauer

"No Freeman shall ever be debarred the use of arms. The strongest
reason for the people to retain the right to keep and bear arms is,
as a last resort, to protect themselves against tyranny in government."
--Thomas Jefferson

Subject: Re: hardRandNumbGen
Date: 27 Jan 1999 12:05:41 -0500
From: juola@mathcs.duq.edu (Patrick Juola)
Message-ID: <78nh16$lt6$1@quine.mathcs.duq.edu>
References: <36AF363E.45670214@stud.uni-muenchen.de>
Newsgroups: sci.crypt
Lines: 23

In article <36AF363E.45670214@stud.uni-muenchen.de>,
Mok-Kong Shen <mok-kong.shen@stud.uni-muenchen.de> wrote:
>R. Knauer wrote:
>>
>
>> What he fails to appreciate is that there is a fundamental difference
>> between a TRNG and a PRNG. That is because he fails to realize that a
>> crypto-grade random number is characterized by the generation process,
>> not the number itself.
>
>Where is the proof of 'if the generation process is hardware then
>it is crypto-grade, otherwise it is not'??

There isn't such a proof. There *is* a proof that if the generation
process is purely software running on a deterministic routine, then
the randomness of the resulting pad is bounded by the randomness
of the starting conditions.

And there's a similar proof that bounded randomness cannot provide
perfect secrecy to a message of unbounded length.

-kitten

Subject: Re: hardRandNumbGen
Date: Wed, 27 Jan 1999 18:29:34 +0100
From: Mok-Kong Shen <mok-kong.shen@stud.uni-muenchen.de>
Message-ID: <36AF4CFE.6AF01030@stud.uni-muenchen.de>
References: <78nh16$lt6$1@quine.mathcs.duq.edu>
Newsgroups: sci.crypt
Lines: 21

Patrick Juola wrote:
>

> >Where is the proof of 'if the generation process is hardware then
> >it is crypto-grade, otherwise it is not'??
>
> There isn't such a proof. There *is* a proof that if the generation
> process is purely software running on a deterministic routine, then
> the randomness of the resulting pad is bounded by the randomness
> of the starting conditions.
>
> And there's a similar proof that bounded randomness cannot provide
> perfect secrecy to a message of unbounded length.

Entirely true. It is for the same reason that none of the known
(implementable) ciphers can offer absolute security. But that
cetainly doesn't imply that any hardware generator can offer
absolute security. One should also note that even 'crypto-grade'
is itself a fuzzy term.

M. K. Shen

Subject: Re: hardRandNumbGen
Date: Wed, 27 Jan 1999 18:13:20 GMT
From: rcktexas@ix.netcom.com (R. Knauer)
Message-ID: <36af56f3.29664925@nntp.ix.netcom.com>
References: <36AF4CFE.6AF01030@stud.uni-muenchen.de>
Newsgroups: sci.crypt
Lines: 25

On Wed, 27 Jan 1999 18:29:34 +0100, Mok-Kong Shen
<mok-kong.shen@stud.uni-muenchen.de> wrote:

>> And there's a similar proof that bounded randomness cannot provide
>> perfect secrecy to a message of unbounded length.

>Entirely true. It is for the same reason that none of the known
>(implementable) ciphers can offer absolute security. But that
>cetainly doesn't imply that any hardware generator can offer
>absolute security. One should also note that even 'crypto-grade'
>is itself a fuzzy term.

You STILL haven't got it.

The OTP cryptosystem is proveably secure.

Why are you struggling with that?

Bob Knauer

"No Freeman shall ever be debarred the use of arms. The strongest
reason for the people to retain the right to keep and bear arms is,
as a last resort, to protect themselves against tyranny in government."
--Thomas Jefferson

Subject: Re: hardRandNumbGen
Date: Wed, 27 Jan 1999 19:36:17 +0100
From: Mok-Kong Shen <mok-kong.shen@stud.uni-muenchen.de>
Message-ID: <36AF5CA1.59F0F2F4@stud.uni-muenchen.de>
References: <36af56f3.29664925@nntp.ix.netcom.com>
Newsgroups: sci.crypt
Lines: 29

R. Knauer wrote:
>
>
> You STILL haven't got it.
>
> The OTP cryptosystem is proveably secure.
>
> Why are you struggling with that?

I never refuse to acknowledge 'The OTP cryptosystem is proveably
secure'. Only as I said repeatedly that that fact is USELESS for
the practice, i.e. useless for the user of a crypto application,
because he can't get a perfect OTP to give him the 'provable
security'. Neither will he, excepting a mentally illed, demand
absolute security. But he has a right to demand that the security

offered by a system be demonstrated. Simply saying that a hardware generator offers high security is NO convincing argument for him. He has to have some supporting scientific data. Now for a random number sequence I don't know any way to provide such data excepting through statistical tests. And by the discussions till now you also don't know a way. So it is my opinion that using statistical tests at least gives him something concrete for him to form a more or less objective opinion, even though we should tell him that the tests are questionable for this and that reason. This is anyway better than leaving him with nothing to form a (subjective) opinion of the encryption scheme he is using. Have I expressed my arguments clearly?

M. K. Shen

Subject: Re: hardRandNumbGen
Date: Wed, 27 Jan 1999 23:37:37 GMT
From: rcktexas@ix.netcom.com (R. Knauer)
Message-ID: <36afa0ac.48537553@nntp.ix.netcom.com>
References: <36AF5CA1.59F0F2F4@stud.uni-muenchen.de>
Newsgroups: sci.crypt
Lines: 57

On Wed, 27 Jan 1999 19:36:17 +0100, Mok-Kong Shen
<mok-kong.shen@stud.uni-muenchen.de> wrote:

>But he has a right to demand that the security
>offered by a system be demonstrated.

Fine - let him seek a meaningful demonstration then.

>Simply saying that a hardware
>generator offers high security

No one ever said that.

>Now for a random
>number sequence I don't know any way to provide such data excepting
>through statistical tests.

Statistical tests do not prove randomness if they are applied to
finite numbers.

>And by the discussions till now you
>also don't know a way.

That's because there is no way, other than to let the TRNG generate an
infinitely long number.

>So it is my opinion that using statistical
>tests at least gives him something concrete for him to form a more
>or less objective opinion, even though we should tell him that the
>tests are questionable for this and that reason.

"Questionable" is a weasel word for meaningless.

Statistical tests are useless if applied to finite random numbers
generated by a TRNG.

>This is anyway
>better than leaving him with nothing to form a (subjective) opinion
>of the encryption scheme he is using.

If a test is worthless (except to diagnose a major malfunction), how
is it any good?

Actually, it is worse than not using it because it can give a false
sense of confidence.

>Have I expressed my arguments clearly?

Yep, it is clear that you are as confused as ever.

Bob Knauer

"No Freeman shall ever be debarred the use of arms. The strongest
reason for the people to retain the right to keep and bear arms is,
as a last resort, to protect themselves against tyranny in government."
--Thomas Jefferson

Subject: Re: hardRandNumbGen
Date: Thu, 28 Jan 1999 16:38:22 +0100
From: Mok-Kong Shen <mok-kong.shen@stud.uni-muenchen.de>
Message-ID: <36B0846D.5113C58D@stud.uni-muenchen.de>
References: <36afa0ac.48537553@nntp.ix.netcom.com>
Newsgroups: sci.crypt
Lines: 40

R. Knauer wrote:
>
>

> Statistical tests are useless if applied to finite random numbers
> generated by a TRNG.

Are you going to REWITE the whole scientific theory of probability
and statistics?????

After having built a hardware generator, do you test it or simply
'believe' that it is o.k.? If you do test, which tests do you perform?
Do you ever consider that the output can have bias? If yes, what
is a bias and how do you detect the bias and quantify it? What tools
do you have in doing all these if you exclude ALL statistical tools???

Since we have argued so much and in that process also touched PRNGs,
I suppose the following paragraph taken from a brand new book by
O. Goldreich, Modern Cryptography, Probabilistic Proofs and
Pseudorandomness, Springer-Verlag, 1999, may interest you (page 77):

> Thus, pseudorandom generators are efficient (i.e. polynomial-time)
> deterministic programs which expand short randomly selected seeds
> into longer pseudorandom bit sequences, where the latter are
> defined as computationally indistinguishable from truly random
> sequences by efficient (i.e. polynomial-time) algorithms. It
> follows that any efficient randomized algorithms maintains its
> performance when its internal coin tosses are substituted by a
> sequence generated by a pseudorandom generator.

Also interesting is a quotation Goldreich puts at the beginning of
chapter 3 of his book:

> If two objects are indistinguishable, in what sense are they
> different?

Since you so often use the term 'crypto-grade', let me also remind
you that there are cryptologically secure PRNGs, the best known
one being that of BBS (see any textbook on modern cryptography).

M. K. Shen

Subject: Re: hardRandNumbGen
Date: Thu, 28 Jan 1999 23:18:56 GMT
From: rcktexas@ix.netcom.com (R. Knauer)
Message-ID: <36b0f039.10200557@nntp.ix.netcom.com>
References: <36B0846D.5113C58D@stud.uni-muenchen.de>
Newsgroups: sci.crypt
Lines: 20

On Thu, 28 Jan 1999 16:38:22 +0100, Mok-Kong Shen
<mok-kong.shen@stud.uni-muenchen.de> wrote:

>> Statistical tests are useless if applied to finite random numbers
>> generated by a TRNG.

>Are you going to REWITE the whole scientific theory of probability
>and statistics?????

I give up. Believe what you want.

You are beyond redemption.

Bob Knauer

"No Freeman shall ever be debarred the use of arms. The strongest
reason for the people to retain the right to keep and bear arms is,
as a last resort, to protect themselves against tyranny in government."
--Thomas Jefferson

Subject: Re: hardRandNumbGen
Date: Wed, 27 Jan 1999 15:30:54 GMT
From: rcktexas@ix.netcom.com (R. Knauer)
Message-ID: <36af2cf5.18914767@nntp.ix.netcom.com>
References: <36AF24E5.95D5D7F9@stud.uni-muenchen.de>
Newsgroups: sci.crypt
Lines: 56

On Wed, 27 Jan 1999 15:38:29 +0100, Mok-Kong Shen
<mok-kong.shen@stud.uni-muenchen.de> wrote:

>Excellent! Then tell me HOW to get such a physical device that
>PROVABLY is capable of generating all possible sequences of a given
>finite length equiprobalbly.

By analyzing the design. With correct design one can make a TRNG
random to within an arbitrarily high level of precision.

If the TRNG is certified to produce crypto-grade random numbers to a
level of precision that ensures that it would take an impossible
amount of work to make it vulnerable, then that is "perfect" in a
practical sense.

If the ball bearings in your car's wheels are spherical enough so that
they do not tear up the bearing races, that is "perfect" enough in a

practical sense.

Being obsessed over the fact that there is no such thing as a Perfect TRNG or a Perfect Sphere in the real world is a waste of time at the practical level. There are more important considerations that affect security - like having someone steal your pad.

One thing is for sure - just because there is no such thing as a Perfect TRNG is no excuse to fall back on PRNGs for the OTP cryptosystem.

>Secondly, your equiprobability is not at all sufficient.

I never said it was. I said that the TRNG must be *CAPABLE* of:

1) Outputting all possible sequences of a given finite length; and

2) Outputting each of them equiprobably.

>If the said given finite length is 2, is a physical device outputting
>0001101100011011..... a TRNG?????

Sure, why not - if you group the number above in 2-bit sequences:

00 01 10 11 00 01 10 11

Pad1: 00
Pad2: 01
Pad3:10
....
Pad8: 11

Bob Knauer

"No Freeman shall ever be debarred the use of arms. The strongest reason for the people to retain the right to keep and bear arms is, as a last resort, to protect themselves against tyranny in government."
--Thomas Jefferson

Subject: Re: hardRandNumbGen
Date: Wed, 27 Jan 1999 16:59:03 +0100
From: Mok-Kong Shen <mok-kong.shen@stud.uni-muenchen.de>
Message-ID: <36AF37C7.EC548C1@stud.uni-muenchen.de>
References: <36af2cf5.18914767@nntp.ix.netcom.com>
Newsgroups: sci.crypt
Lines: 27

R. Knauer wrote:
>

> By analyzing the design. With correct design one can make a TRNG
> random to within an arbitrarily high level of precision.
>
> If the TRNG is certified to produce crypto-grade random numbers to a
> level of precision that ensures that it would take an impossible
> amount of work to make it vulnerable, then that is "perfect" in a
> practical sense.
>
> If the ball bearings in your car's wheels are spherical enough so that
> they do not tear up the bearing races, that is "perfect" enough in a
> practical sense.
>
> Being obsessed over the fact that there is no such thing as a Perfect
> TRNG or a Perfect Sphere in the real world is a waste of time at the
> practical level. There are more important considerations that affect
> security - like having someone steal your pad.

If you make a metal sphere, there is a common definition of precision.
What is the 'precision' you are referring to about your TRNG design?
You have to define that 'precision' in scientific terms, in particular
establish a 'unit' and provide a precise method to measure that
'precision' in that unit. Before that, you have nothing.

M. K. Shen

Subject: Re: hardRandNumbGen
Date: Wed, 27 Jan 1999 16:31:33 GMT
From: rcktexas@ix.netcom.com (R. Knauer)
Message-ID: <36af3c48.22837718@nntp.ix.netcom.com>
References: <36AF37C7.EC548C1@stud.uni-muenchen.de>
Newsgroups: sci.crypt
Lines: 42

On Wed, 27 Jan 1999 16:59:03 +0100, Mok-Kong Shen
<mok-kong.shen@stud.uni-muenchen.de> wrote:

>If you make a metal sphere, there is a common definition of precision.
>What is the 'precision' you are referring to about your TRNG design?
>You have to define that 'precision' in scientific terms, in particular
>establish a 'unit' and provide a precise method to measure that
>'precision' in that unit. Before that, you have nothing.

You do that by analyzing the design, and performing certain tests on
the actual equipment to make sure it meets the design specifications.

One such test for a radioactive decay TRNG would be to measure the
radioactive decay before the interval measuring circuitry. If the
measurements yields results that you expect, there is no reason to
believe anything is wrong.

As far as the digital circuitry is concerned, logic analysis is what
you would use. You would put the digital circuits on a logic analyzer
and certify that they perform to design specifications using simulated
inputs. You would inject noise into the components and see if it had a
measureable effect on the output.

You are relying on two facts:

1) Quantum mechanical processes result in random events - otherwise
physics wouldn't work;

2) Digital circuits are incredibly robust - otherwise computers
wouldn't work.

You can bet the NSA has a TRNG that is certified to be completely
secure in a practical sense, where the work effort to break it would
be more than the energy in the Universe.

Bob Knauer

"No Freeman shall ever be debarred the use of arms. The strongest
reason for the people to retain the right to keep and bear arms is,
as a last resort, to protect themselves against tyranny in government."
--Thomas Jefferson

Subject: Re: hardRandNumbGen
Date: Wed, 27 Jan 1999 17:55:15 +0100
From: Mok-Kong Shen <mok-kong.shen@stud.uni-muenchen.de>
Message-ID: <36AF44F3.6132574A@stud.uni-muenchen.de>
References: <36af3c48.22837718@nntp.ix.netcom.com>
Newsgroups: sci.crypt
Lines: 36

R. Knauer wrote:
>
> On Wed, 27 Jan 1999 16:59:03 +0100, Mok-Kong Shen
> <mok-kong.shen@stud.uni-muenchen.de> wrote:
>
> >If you make a metal sphere, there is a common definition of precision.
> >What is the 'precision' you are referring to about your TRNG design?
> >You have to define that 'precision' in scientific terms, in particular
> >establish a 'unit' and provide a precise method to measure that
> >'precision' in that unit. Before that, you have nothing.
>
> You do that by analyzing the design, and performing certain tests on
> the actual equipment to make sure it meets the design specifications.
>
> One such test for a radioactive decay TRNG would be to measure the
> radioactive decay before the interval measuring circuitry. If the
> measurements yields results that you expect, there is no reason to
> believe anything is wrong.
>
> As far as the digital circuitry is concerned, logic analysis is what
> you would use. You would put the digital circuits on a logic analyzer
> and certify that they perform to design specifications using simulated
> inputs. You would inject noise into the components and see if it had a
> measureable effect on the output.

It is important to know what the specifications ARE. Certainly
things like the dimensions of the apparatus don't have too much
bearing in the present context. Now what are the specifications that
ensures a crypto-grade TRNG? These specifications must contain
certain numerical values in terms of certain units. Then one can
test the actual product to see whether the specifications are
fulfilled. As long as one can't define 'crypto-grade' in terms
of certain units precisely, there is NO way to write up such
specifications as you proposed.

M. K. Shen

Subject: Re: hardRandNumbGen
Date: Wed, 27 Jan 1999 17:39:15 GMT
From: rcktexas@ix.netcom.com (R. Knauer)
Message-ID: <36af4ec9.27574880@nntp.ix.netcom.com>
References: <36AF44F3.6132574A@stud.uni-muenchen.de>
Newsgroups: sci.crypt
Lines: 24

On Wed, 27 Jan 1999 17:55:15 +0100, Mok-Kong Shen
<mok-kong.shen@stud.uni-muenchen.de> wrote:

>It is important to know what the specifications ARE. Certainly

>things like the dimensions of the apparatus don't have too much
>bearing in the present context. Now what are the specifications that
>ensures a crypto-grade TRNG? These specifications must contain
>certain numerical values in terms of certain units. Then one can
>test the actual product to see whether the specifications are
>fulfilled. As long as one can't define 'crypto-grade' in terms
>of certain units precisely, there is NO way to write up such
>specifications as you proposed.

Check out the Hotbits radioactive decay TRNG:

http://www.fourmilab.ch/hotbits/

Bob Knauer

"No Freeman shall ever be debarred the use of arms. The strongest
reason for the people to retain the right to keep and bear arms is,
as a last resort, to protect themselves against tyranny in government."
--Thomas Jefferson

Subject: Re: hardRandNumbGen
Date: 27 Jan 1999 13:13:21 -0500
From: juola@mathcs.duq.edu (Patrick Juola)
Message-ID: <78nl01$m0t$1@quine.mathcs.duq.edu>
References: <36AF44F3.6132574A@stud.uni-muenchen.de>
Newsgroups: sci.crypt
Lines: 66

In article <36AF44F3.6132574A@stud.uni-muenchen.de>,
Mok-Kong Shen <mok-kong.shen@stud.uni-muenchen.de> wrote:
>R. Knauer wrote:
>>
>> On Wed, 27 Jan 1999 16:59:03 +0100, Mok-Kong Shen
>> <mok-kong.shen@stud.uni-muenchen.de> wrote:
>>
>> >If you make a metal sphere, there is a common definition of precision.
>> >What is the 'precision' you are referring to about your TRNG design?
>> >You have to define that 'precision' in scientific terms, in particular
>> >establish a 'unit' and provide a precise method to measure that
>> >'precision' in that unit. Before that, you have nothing.
>>
>> You do that by analyzing the design, and performing certain tests on
>> the actual equipment to make sure it meets the design specifications.
>>
>> One such test for a radioactive decay TRNG would be to measure the
>> radioactive decay before the interval measuring circuitry. If the
>> measurements yields results that you expect, there is no reason to
>> believe anything is wrong.
>
>It is important to know what the specifications ARE. Certainly
>things like the dimensions of the apparatus don't have too much
>bearing in the present context. Now what are the specifications that
>ensures a crypto-grade TRNG?

Well, broadly speaking, you need a source of randomness, which will
presumably be specified. As in the specification for ball bearings,
your specification will not cover all possible configurations, but
only one that you consider acceptable -- for example, you might
specify that your bearing be made of steel when brass bearings *might*
also be acceptable.

So I'll specify that the source of randomness is a particular mass
of depleted uranium of such-and-such purity. Why did I specify
this? I believe -- and I have lots of evidence from physicists to
support such a belief -- that radioactive events in such and such
a mass are "truly random."

Then, I need to sample those events in a manner that preserves the
randomness. Von Neumann's math provides a formal specification of
a process to do so -- or I could use a decent hash scheme or any
of several alternatives to specify the "formal" properties of such
a system. Once I've got the formal requirements, building such
a gadget (and certifying its compliance to the specs) is just
another engineering problem.

Of course, depending on the sampling techniques I specify, I could
come up with many different -- and all equally effective -- gadget
designs. Or I could make the ball bearings out of aluminum....

Once I've done that, I will presumably need to specify some form
of (formalizable) security to make sure that the whole thing is
wrapped in a shielded and tamper-evident package. Another
formal spec, implemented by an engineer.

The resulting number streams can be certified as "random," subject
to the same (standard) assumptions that any other engineering project
uses.

Note : I'm not claiming that this particular gizmo is the *only* way
to get crypto-strength random numbers. Perhaps what you want for
your random number source is a thorium source, or Brownian motion.
But anything you build according to these gizmo specs can be certified.

-kitten

Subject: Re: hardRandNumbGen
Date: Wed, 27 Jan 1999 12:02:57 -0500
From: "Trevor Jackson, III" <fullmoon@aspi.net>
Message-ID: <36AF46C1.8FA0A086@aspi.net>
References: <36AF37C7.EC548C1@stud.uni-muenchen.de>
Newsgroups: sci.crypt
Lines: 33

Mok-Kong Shen wrote:

> R. Knauer wrote:
> >
>
> > By analyzing the design. With correct design one can make a TRNG
> > random to within an arbitrarily high level of precision.
> >
> > If the TRNG is certified to produce crypto-grade random numbers to a
> > level of precision that ensures that it would take an impossible
> > amount of work to make it vulnerable, then that is "perfect" in a
> > practical sense.
> >
> > If the ball bearings in your car's wheels are spherical enough so that
> > they do not tear up the bearing races, that is "perfect" enough in a
> > practical sense.
> >
> > Being obsessed over the fact that there is no such thing as a Perfect
> > TRNG or a Perfect Sphere in the real world is a waste of time at the
> > practical level. There are more important considerations that affect
> > security - like having someone steal your pad.
>
> If you make a metal sphere, there is a common definition of precision.
> What is the 'precision' you are referring to about your TRNG design?
> You have to define that 'precision' in scientific terms, in particular
> establish a 'unit' and provide a precise method to measure that
> 'precision' in that unit. Before that, you have nothing.

Now this is an issue worthy of intense thought and debate (emphasis on
thought please). I believe this breaks into two subtopics, one
fundamentally describing the unit of measure and the other describing the
measurement methodology.

Subject: Re: hardRandNumbGen
Date: Wed, 27 Jan 1999 12:31:11 -0600
From: Medical Electronics Lab <rosing@physiology.wisc.edu>
Message-ID: <36AF5B6F.6C2B@physiology.wisc.edu>
References: <36AF37C7.EC548C1@stud.uni-muenchen.de>
Newsgroups: sci.crypt
Lines: 24

Mok-Kong Shen wrote:
>
> If you make a metal sphere, there is a common definition of precision.
> What is the 'precision' you are referring to about your TRNG design?
> You have to define that 'precision' in scientific terms, in particular
> establish a 'unit' and provide a precise method to measure that
> 'precision' in that unit. Before that, you have nothing.

Usually it's chi-square and KS-tests which output a probability
parameter (usually called "p"). Precision of more than 3 digits
isn't really necessary, either the p value is within a reasonable
range (+/- 1.5 sigma from some mean) or it isn't. testing multiple
blocks of data of the same length will give different p's (or you
know something's crooked!) but they should all be with in the
correct range.

The "unit" is expectation value. Precision isn't a good term,
what you want is "uniformity". A TRNG should have expectation
values for *all* imaginable tests, i.e. it should be uniformly
random no matter how you look at it. While not easy to do, it
is possible with careful attention to details.

Patience, persistence, truth,
Dr. mike

Subject: Re: hardRandNumbGen
Date: 27 Jan 1999 12:01:50 -0500
From: juola@mathcs.duq.edu (Patrick Juola)
Message-ID: <78ngpulrm1@quine.mathcs.duq.edu>
References: <36af2cf5.18914767@nntp.ix.netcom.com>
Newsgroups: sci.crypt
Lines: 43

In article <36af2cf5.18914767@nntp.ix.netcom.com>,
R. Knauer <rcktexas@ix.netcom.com> wrote:
>On Wed, 27 Jan 1999 15:38:29 +0100, Mok-Kong Shen
><mok-kong.shen@stud.uni-muenchen.de> wrote:

>
>>Secondly, your equiprobability is not at all sufficient.
>
>I never said it was. I said that the TRNG must be *CAPABLE* of:
>
>1) Outputting all possible sequences of a given finite length; and
>
>2) Outputting each of them equiprobably.
>
>>If the said given finite length is 2, is a physical device outputting
>>0001101100011011..... a TRNG?????
>
>Sure, why not - if you group the number above in 2-bit sequences:
>
>00 01 10 11 00 01 10 11
>
>Pad1: 00
>Pad2: 01
>Pad3:10
>....
>Pad8: 11

Minor quantification error :

Mr. Knauer's criterion should not be interpreted as:

There exists a length such that all possible sequences of that
length are outputted equiprobably,

but as :

For all (finite) lengths, all possible sequences of that length are
outputted equiprobable.

So if that's really the output of your generator, it's *NOT* a TRNG
as it's a single 8-bit sequence repeated with probability 1.

-kitten

Subject: Re: hardRandNumbGen
Date: Wed, 27 Jan 1999 18:10:02 GMT
From: rcktexas@ix.netcom.com (R. Knauer)
Message-ID: <36af54d6.29123497@nntp.ix.netcom.com>
References: <78ngpulrm1@quine.mathcs.duq.edu>
Newsgroups: sci.crypt
Lines: 56

On 27 Jan 1999 12:01:50 -0500, juola@mathcs.duq.edu (Patrick Juola)
wrote:

>>>Secondly, your equiprobability is not at all sufficient.

>>I never said it was. I said that the TRNG must be *CAPABLE* of:

>>1) Outputting all possible sequences of a given finite length; and

>>2) Outputting each of them equiprobably.

>>>If the said given finite length is 2, is a physical device outputting
>>>0001101100011011..... a TRNG?????

>>Sure, why not - if you group the number above in 2-bit sequences:
>>
>>00 01 10 11 00 01 10 11
>>
>>Pad1: 00
>>Pad2: 01
>>Pad3:10
>>....
>>Pad8: 11
>
>
>Minor quantification error :
>
>Mr. Knauer's criterion should not be interpreted as:
>
>There exists a length such that all possible sequences of that
>length are outputted equiprobably,
>
>but as :
>
>For all (finite) lengths, all possible sequences of that length are
>outputted equiprobable.
>
>So if that's really the output of your generator, it's *NOT* a TRNG
>as it's a single 8-bit sequence repeated with probability 1.

I assumed the poster had generated the 16 bit sequence above using a
TRNG, like a fair coin toss. I then assumed that he wanted to use it
to make a pad. Then he says something about a length of 2, which I
assumed he meant as message lengths - that he planned to send 8
messages of 2 bits in length.

Given all those assumptions, I see no problem with using the sequence
above as for an OTP - unless I am missing something.

Bob Knauer

"No Freeman shall ever be debarred the use of arms. The strongest
reason for the people to retain the right to keep and bear arms is,
as a last resort, to protect themselves against tyranny in government."
--Thomas Jefferson

Subject: Re: hardRandNumbGen
Date: Fri, 29 Jan 1999 05:48:38 GMT
From: cr764@torfree.net (Kurt Wismer)
Message-ID: <F6B453.D3o.0.queen@torfree.net>
References: <36ae16a8.29884341@nntp.ix.netcom.com>
Newsgroups: sci.crypt
Lines: 19

R. Knauer (rcktexas@ix.netcom.com) wrote:
: Learn what crypto-grade randomness is. The concept is deceptively
: simple once you understand it. But first you have to give up all other
: definitions of randomness from other fields like statistics.

: The key to understanding is that randomness depends on the generation
: process, not the numbers themselves. The number 000...0 fails all
: sorts of statistical tests, but can be a random number if it is
: generated by a TRNG. Until you analyze the method of generation, you
: cannot know.

this is the definition i've used for years... strangely, nothing i ever
learned in statistics ever suggested i was wrong...

--
"some speak the sounds but speak in silent voices
 like radio is silent though it fills the air with noises
 its transmissions bring submission as ya mold to the unreal
 mad boy grips the microphone wit' a fistful of steel"

Subject: Re: hardRandNumbGen
Date: 29 Jan 1999 09:12:24 -0500
From: juola@mathcs.duq.edu (Patrick Juola)
Message-ID: <78sfk8$ou7$1@quine.mathcs.duq.edu>
References: <F6B453.D3o.0.queen@torfree.net>
Newsgroups: sci.crypt
Lines: 25

In article <F6B453.D3o.0.queen@torfree.net>,
Kurt Wismer <cr764@torfree.net> wrote:
>R. Knauer (rcktexas@ix.netcom.com) wrote:
>: Learn what crypto-grade randomness is. The concept is deceptively
>: simple once you understand it. But first you have to give up all other
>: definitions of randomness from other fields like statistics.
>
>: The key to understanding is that randomness depends on the generation
>: process, not the numbers themselves. The number 000...0 fails all
>: sorts of statistical tests, but can be a random number if it is
>: generated by a TRNG. Until you analyze the method of generation, you
>: cannot know.
>
>this is the definition i've used for years... strangely, nothing i ever
>learned in statistics ever suggested i was wrong...

Possibly because it isn't. 8-)

On the other hand, it also looks suspiciously like the result of
an incompetent engineer *trying* to build a RNG.

Which is it? Your call. You know the engineers that you hired....

        -kitten

Subject: Re: hardRandNumbGen
Date: Sat, 30 Jan 1999 17:15:11 GMT
From: cr764@torfree.net (Kurt Wismer)
Message-ID: <F6DuLB.Lun.0.queen@torfree.net>
References: <78sfk8$ou7$1@quine.mathcs.duq.edu>
Newsgroups: sci.crypt
Lines: 43

Patrick Juola (juola@mathcs.duq.edu) wrote:
: In article <F6B453.D3o.0.queen@torfree.net>,
: Kurt Wismer <cr764@torfree.net> wrote:
: >R. Knauer (rcktexas@ix.netcom.com) wrote:
: >: Learn what crypto-grade randomness is. The concept is deceptively
: >: simple once you understand it. But first you have to give up all other
: >: definitions of randomness from other fields like statistics.
: >

: >: The key to understanding is that randomness depends on the generation
: >: process, not the numbers themselves. The number 000...0 fails all
: >: sorts of statistical tests, but can be a random number if it is
: >: generated by a TRNG. Until you analyze the method of generation, you
: >: cannot know.
: >
: >this is the definition i've used for years... strangely, nothing i ever
: >learned in statistics ever suggested i was wrong...

: Possibly because it isn't. 8-)

: On the other hand, it also looks suspiciously like the result of
: an incompetent engineer *trying* to build a RNG.

: Which is it? Your call. You know the engineers that you hired....

the null output you mean?

i can't know simply by looking at the output... statistical tests can be diagnostic and suggest whether bias was unintentionally introduced... if the generator is relatively cheap i might ask the engineers (or maybe a different group of engineers) to build another to see if it behaves similarly... might also want to have group go over the first one with a fine tooth comb to make sure it meets design specifications...

it may be that the design is prone to errors in implementation, in which case it should be redesigned (at least if the rng is meant for large scale production)... it might also be a statistical anomaly... i don't see any foolproof method of verifying that the trng is indeed a trng, however...
--
"some speak the sounds but speak in silent voices
 like radio is silent though it fills the air with noises
 its transmissions bring submission as ya mold to the unreal
 mad boy grips the microphone wit' a fistful of steel"

Subject: Re: hardRandNumbGen
Date: Sun, 31 Jan 1999 00:08:31 GMT
From: rcktexas@ix.netcom.com (R. Knauer)
Message-ID: <36b39ec6.42742981@nntp.ix.netcom.com>
References: <F6DuLB.Lun.0.queen@torfree.net>
Newsgroups: sci.crypt
Lines: 17

On Sat, 30 Jan 1999 17:15:11 GMT, cr764@torfree.net (Kurt Wismer)
wrote:

>i don't
>see any foolproof method of verifying that the trng is indeed a trng,
>however...

I believe that a radioactive decay TRNG can be verified to within a
known level of precision.

Bob Knauer

"I place economy among the first and most important virtues and
public debt as the greatest dangers to be feared. We must not
let our rulers load us with perpetual debt."
--Thomas Jefferson

Subject: Re: hardRandNumbGen
Date: 1 Feb 1999 09:18:14 -0500
From: juola@mathcs.duq.edu (Patrick Juola)
Message-ID: <794d36$qh0$1@quine.mathcs.duq.edu>
References: <36b39ec6.42742981@nntp.ix.netcom.com>
Newsgroups: sci.crypt
Lines: 18

In article <36b39ec6.42742981@nntp.ix.netcom.com>,
R. Knauer <rcktexas@ix.netcom.com> wrote:
>On Sat, 30 Jan 1999 17:15:11 GMT, cr764@torfree.net (Kurt Wismer)
>wrote:
>
>>i don't
>>see any foolproof method of verifying that the trng is indeed a trng,
>>however...
>
>I believe that a radioactive decay TRNG can be verified to within a
>known level of precision.

But this is yet another example of a statistical test. There will be
some level of precision greater than which you can't test -- and
some possiblity that the randomness will result in an apparent
aberration.

        -kitten

Subject: Re: hardRandNumbGen

Date: Wed, 03 Feb 1999 18:44:56 GMT
From: rcktexas@ix.netcom.com (R. Knauer)
Message-ID: <36b8955c.18282849@nntp.ix.netcom.com>
References: <794d36$qh0$1@quine.mathcs.duq.edu>
Newsgroups: sci.crypt
Lines: 44

On 1 Feb 1999 09:18:14 -0500, juola@mathcs.duq.edu (Patrick Juola)
wrote:

>>I believe that a radioactive decay TRNG can be verified to within a
>>known level of precision.

>But this is yet another example of a statistical test. There will be
>some level of precision greater than which you can't test -- and
>some possiblity that the randomness will result in an apparent
>aberration.

I covered this point in an earlier post. The tests, statistical or
otherwise, would be of a diagnostic nature applied to the sybsystems
of the TRNG. Those diagnostics would be based on the design of the
applicable subsystem in terms of known modes of malfunction. Therefore
those diagnostic tests are of a determinant nature since their results
can be related to a known condition.

That cannot be said of statistical tests of the final output sequence,
which by definition is completely indeterminant. Of course, something
in one of the subsystems of the TRNG must be random, and testing it
statistically would not be valid. But the fact that it is random can
be inferred from the nature of the underlying physical process, like
with radioactive decay.

Therefore a complete audit of the TRNG, subsystem by subsystem, can be
conducted resulting in a known level of precision for the final
output. Such a TRNG can be certified as proveably secure if the level
of precision is such that it would take an impossibly large work
effort to decrypt OTP ciphers.

The problem with statistical tests on the final output is that there
is no reliable way to quantify the level of precision for ALL outputs,
and there is no reliable way to filter out "bad" outputs since there
is no such thing as a "bad" output - except possibly in the case of
the diagnostic test for all 1s (open output) or all 0s (shorted
output).

Bob Knauer

"Sometimes it is said that man cannot be trusted with the government
of himself. Can he, then, be trusted with the government of others?"
--Thomas Jefferson

Subject: Re: hardRandNumbGen
Date: Wed, 03 Feb 1999 16:08:03 -0500
From: "Trevor Jackson, III" <fullmoon@aspi.net>
Message-ID: <36B8BAB2.5D8D2F7E@aspi.net>
References: <36b8955c.18282849@nntp.ix.netcom.com>
Newsgroups: sci.crypt
Lines: 49

R. Knauer wrote:

> On 1 Feb 1999 09:18:14 -0500, juola@mathcs.duq.edu (Patrick Juola)
> wrote:
>
> >>I believe that a radioactive decay TRNG can be verified to within a
> >>known level of precision.
>
> >But this is yet another example of a statistical test. There will be
> >some level of precision greater than which you can't test -- and
> >some possiblity that the randomness will result in an apparent
> >aberration.
>
> I covered this point in an earlier post. The tests, statistical or
> otherwise, would be of a diagnostic nature applied to the sybsystems
> of the TRNG. Those diagnostics would be based on the design of the
> applicable subsystem in terms of known modes of malfunction. Therefore
> those diagnostic tests are of a determinant nature since their results
> can be related to a known condition.

So a properly designed RNG is not permitted to fails in an unknown way?
Is this design methodology documented anywhere? It sounds like it avoids
Murphy the way perpetual motion machines avoid friction.

>
>
> That cannot be said of statistical tests of the final output sequence,
> which by definition is completely indeterminant. Of course, something
> in one of the subsystems of the TRNG must be random, and testing it
> statistically would not be valid. But the fact that it is random can
> be inferred from the nature of the underlying physical process, like
> with radioactive decay.
>

> Therefore a complete audit of the TRNG, subsystem by subsystem, can be
> conducted resulting in a known level of precision for the final
> output. Such a TRNG can be certified as proveably secure if the level
> of precision is such that it would take an impossibly large work
> effort to decrypt OTP ciphers.
>
> The problem with statistical tests on the final output is that there
> is no reliable way to quantify the level of precision for ALL outputs,
> and there is no reliable way to filter out "bad" outputs since there
> is no such thing as a "bad" output - except possibly in the case of
> the diagnostic test for all 1s (open output) or all 0s (shorted
> output).

Now I get it. We shouldn't test because there might be a source of
entropy present other than that which is designed to be present.

Subject: Re: hardRandNumbGen
Date: 27 Jan 1999 08:37:36 -0500
From: juola@mathcs.duq.edu (Patrick Juola)
Message-ID: <78n4r0$l6l$1@quine.mathcs.duq.edu>
References: <36AE0852.16B9A95F@stud.uni-muenchen.de>
Newsgroups: sci.crypt
Lines: 55

In article <36AE0852.16B9A95F@stud.uni-muenchen.de>,
Mok-Kong Shen <mok-kong.shen@stud.uni-muenchen.de> wrote:
>R. Knauer wrote:
>>
>> On Tue, 26 Jan 1999 13:33:49 +0100, Mok-Kong Shen
>> <mok-kong.shen@stud.uni-muenchen.de> wrote:
>>
>> >If I have two sources of randomness, one software and one hardware,
>> >both passing all statistical tests I apply equally well, why should
>> >I choose one source in preference to the other?
>>
>> Why do you persist in believing that statistical tests have anything
>> to do with randomness in cryptography?
>
>Tell me what other (better) tools are available for me to make
>the decision. These are simply easy to obtain, as far as my
>humble knowledge goes. Please kindly give your recipe to cope with
>the situation I described. Thanks in advance.

There's an old joke about a man losing a quarter and looking for
it on the wrong corner "because the light is better here." Or
as another metaphor, if I have two used cars, one a Chevrolet and
one a Ford, both with identical option packages, which one's the
better buy?

Only an idiot buys a used car on the basis of the option package;
anyone with any sense will look under the hood, test drive it, and
so forth. And if you know nothing at all about engines, if you're
sensible you'll get a knowledgeable friend or a professional mechanic
to look at it and figure out whether or not it's got some lurking
evil hiding in the clutch master cylinder. (No, I'm not bitter about
that old Renault. Not at all!)

You *can't* evaluate a random number generator based only on the
statistical properties of the stream it produces. Yes, we've got
lots of useful statistical tools that will tell you if it's producing
an obviously biased stream, and if it can't pass them, then it's
not very good.

BUT that's not enough. You are going to have to open the hood and
look at the engine and see how it's put together. Lots of things
that *look* random -- LFSRs, congruential generators, "chaotic function"
generators -- have been cracked. Lots of apparent hardware randomness
solutions have subtle, or not so subtle, biases. Lots of people have
written lots of buggy code.

At this level of evaluation, there are no *tests* per se, although
there are lots of informal checks. E.g., if someone uses a PRNG
with a 32-bit seed, that's not random enough. If someone uses
a pure LFSR, it doesn't matter *how* large the seed is, it's not
secure. And so forth. But the idea that if you can't run a formal
test or if there isn't a software package to do the work for you,
it doesn't matter, is pure and unadulterated bullshit.

-kitten

Subject: Re: hardRandNumbGen
Date: Wed, 27 Jan 1999 14:54:25 GMT
From: rcktexas@ix.netcom.com (R. Knauer)
Message-ID: <36af1f4b.15416708@nntp.ix.netcom.com>
References: <78n4r0$l6l$1@quine.mathcs.duq.edu>
Newsgroups: sci.crypt
Lines: 42

On 27 Jan 1999 08:37:36 -0500, juola@mathcs.duq.edu (Patrick Juola)
wrote:

>You *can't* evaluate a random number generator based only on the
>statistical properties of the stream it produces.

I fully agree for finite sequences, but as I mention in the thread entitled "Metaphysics of Randomness", statistical tests presumably do apply to infinite sequences. But then, an infinite sequence contains all the details about the generation process, so that makes sense.

On an earlier occasion I asked if the Bayesian attack could be used to characterize crypto-grade random numbers. If I gave you a large number of ciphers of substantial length based on some unknown key generation procedure, you tried the Bayesian attack on them and they "leaked" no significant amount of information, then you would probably conclude that the pads were most likely generated by a TRNG, even if I had used some other method like a well-hashed stream cipher. Then I went on to ask if such a "Bayesian Test" for vulnerability could be used to produce confidence limits on the security of subsequent ciphers produced by that encryption method.

Assuming that you could use such a Bayesian Test to characterize randomness, how does that differ from statistical tests for randomness? If you were to take the combined keys for all the ciphers used in the Bayesian Test and concatenate them to one very large number for statistical testing, we know that does not tell us if the number is random - otherwise statistical testing could be used to characterize randomness.

But why should applying the Bayesian Test to presumably random numbers be any different? (I think I know why - one is a statistical test, the other is a probability survey.)

Can you or anyone else please comment on this.

Bob Knauer

"An honest man can feel no pleasure in the exercise of power over his fellow citizens."
--Thomas Jefferson

Subject: Re: hardRandNumbGen
Date: Wed, 27 Jan 1999 16:16:24 +0100
From: Mok-Kong Shen <mok-kong.shen@stud.uni-muenchen.de>
Message-ID: <36AF2DC8.197D196@stud.uni-muenchen.de>
References: <36af1f4b.15416708@nntp.ix.netcom.com>
Newsgroups: sci.crypt
Lines: 10

R. Knauer wrote:
>

> But why should applying the Bayesian Test to presumably random numbers
> be any different? (I think I know why - one is a statistical test, the
> other is a probability survey.)

What is a 'probability survey'? Any literature reference?

M. K. Shen

Subject: Re: hardRandNumbGen
Date: Wed, 27 Jan 1999 15:53:39 GMT
From: rcktexas@ix.netcom.com (R. Knauer)
Message-ID: <36af363c.21289092@nntp.ix.netcom.com>
References: <36AF2DC8.197D196@stud.uni-muenchen.de>
Newsgroups: sci.crypt
Lines: 18

On Wed, 27 Jan 1999 16:16:24 +0100, Mok-Kong Shen
<mok-kong.shen@stud.uni-muenchen.de> wrote:

>What is a 'probability survey'? Any literature reference?

I used the term to describe the process whereby one surveyed a large number of ciphers using probability techniques such as the Bayesian method.

See Patrick's following comments.

Bob Knauer

"No Freeman shall ever be debarred the use of arms. The strongest reason for the people to retain the right to keep and bear arms is, as a last resort, to protect themselves against tyranny in government."
--Thomas Jefferson

Subject: Re: hardRandNumbGen
Date: 27 Jan 1999 10:21:56 -0500
From: juola@mathcs.duq.edu (Patrick Juola)
Message-ID: <78nauklks1@quine.mathcs.duq.edu>

References: <36af1f4b.15416708@nntp.ix.netcom.com>
Newsgroups: sci.crypt
Lines: 113

In article <36af1f4b.15416708@nntp.ix.netcom.com>,
R. Knauer <rcktexas@ix.netcom.com> wrote:
>On 27 Jan 1999 08:37:36 -0500, juola@mathcs.duq.edu (Patrick Juola)
>wrote:
>
>>You *can't* evaluate a random number generator based only on the
>>statistical properties of the stream it produces.
>
>I fully agree for finite sequences, but as I mention in the thread
>entitled "Metaphysics of Randomness", statistical tests presumably do
>apply to infinite sequences. But then, an infinite sequence contains
>all the details about the generation process, so that makes sense.

Umm.... this is a very limited use of the word "test." I will, of course, be delighted to apply any test you choose to any sequence you choose, finite or note. But I charge by the hour, and it would, of course, be improper of me to give you any results until after I've examined the entire sequence.

Still want me to test that infinite sequence for you?

>On an earlier occasion I asked if the Bayesian attack could be used to
>characterize crypto-grade random numbers. If I gave you a large number
>of ciphers of substantial length based on some unknown key generation
>procedure, you tried the Bayesian attack on them and they "leaked" no
>significant amount of information, then you would probably conclude
>that the pads were most likely generated by a TRNG, even if I had used
>some other method like a well-hashed stream cipher.

Probably. By the time you get up to the upper reaches of what we can do with PRNGs, the information leak over reasonably-sizes samples is pretty infinitesimal. Having only reasonably-sized amounts of funding, computing power, and patience at my disposal, there's a limit to what can be detected -- and if I get something that's indistinguishable from random by this test, then I'm probably willing to pass it as a "good" RNG.

On the other hand, there's definitely a possiblity of error there, which is why I wouldn't be likely to make any such statement without asking to see how you generated the key sequence(s). An example of something you could "sneak under the radar" would be a very large LFSR. There are well-known tests of linear complexity that give the minimum size of an LFSR to generate a given sequence. A "truly random" sequence will (with probability 1) yield a complexity curve looking something like this :

```
          _/
        _/
      _/
    _/
  _/
 _/
/
```

A LFSR-generated sequence will have a complexity curve something like this :

```
        ____________________________
      _/
    _/
  _/
 _/
/
```

where it levels off after you hit the generator size.

The question, however, is

a) did you give me enough data to find where the sequence levels off?
b) did I run a test powerful enough and long enough to find that point?

The first is a mathematical property of the sequence. The second is a question of my time, expertise, and a bit of luck.

The problem is that the Bayesian attack only works if I know -- or can guess -- the type of bias likely to be in your generator. In most cases this isn't a problem; many biases are blatantly obvious or result from common mistakes. The more sophisticated you are at hiding your biases, the less likely I am to guess the sort of bias or to lose patience/funding before I complete the necessary tests. (Of course, if I have an infinite amount of time/money, I can make an infinite number of guesses and will probably guess right on at least one of them.)

On the other hand, if I can get a copy of your code, I can just read the code and determine your biases. But you can't rely on keeping your code secret....

> Then I went on to
>ask if such a "Bayesian Test" for vulnerability could be used to

>produce confidence limits on the security of subsequent ciphers
>produced by that encryption method.

In theory, given a large enough number of computers, without question. But we're running into serious funding difficulties here.

>But why should applying the Bayesian Test to presumably random numbers
>be any different? (I think I know why - one is a statistical test, the
>other is a probability survey.)

More a question about finite vs. infinite data. Bayes' Theorem lets you refine hypotheses about biases that you've already made. Conventional statistics just let you test for the presence or absence of a visible bias. As any statistician will tell you, you can't prove the absence of an effect by statistical means. You can just prove that it didn't show up in your experiment and therefore was less than the sensitivity of your test.

Of course, with infinite data, you can develop "tests," in the loosest possible sense, of infinite sensitivity. But this isn't helpful.

-kitten

Subject: Re: hardRandNumbGen
Date: Wed, 27 Jan 1999 16:12:42 GMT
From: rcktexas@ix.netcom.com (R. Knauer)
Message-ID: <36af34e2.20943054@nntp.ix.netcom.com>
References: <78nauklks1@quine.mathcs.duq.edu>
Newsgroups: sci.crypt
Lines: 97

On 27 Jan 1999 10:21:56 -0500, juola@mathcs.duq.edu (Patrick Juola)
wrote:

>But I charge by the hour, and it would,
>of course, be improper of me to give you any results until after I've
>examined the entire sequence.

>Still want me to test that infinite sequence for you?

Sure! As long as I only have to pay you when the test is completed.
:-)

>Probably. By the time you get up to the upper reaches of what we
>can do with PRNGs, the information leak over reasonably-sizes
>samples is pretty infinitesimal.

Is there a way to measure that - to give confidence limits?

>Having only reasonably-sized
>amounts of funding, computing power, and patience at my disposal,
>there's a limit to what can be detected -- and if I get something
>that's indistinguishable from random by this test, then I'm probably
>willing to pass it as a "good" RNG.

I assumed that you had all the existing resources at your disposal
that you wanted - like the NSA has.

>a complexity curve looking
>something like this :
>
> _/
> _/
> _/
> _/
> _/
> _/
>/

What is a "complexity curve" and how do you generate one?

References please, including web sites if possible.

>The problem is that the Bayesian attack only works if I know -- or
>can guess -- the type of bias likely to be in your generator.

So, the Bayesian attack is not all that powerful against stream ciphers *in general*. You have to provide the first hypothesis to get it started. And if you cannot provide a decent hypothesis, then the Bayesian attack is worthless.

>On the other hand, if I can get a copy of your code, I can just read
>the code and determine your biases. But you can't rely on keeping
>your code secret....

I can rely on keeping it just as secret as I keep my keys. If my code has been compromised, so have my keys.

>But we're running into serious funding difficulties here.

Not if you are the NSA. There is no such thing as a funding difficulty when you have access to OPM.

>More a question about finite vs. infinite data. Bayes' Theorem lets you
>refine hypotheses about biases that you've already made. Conventional
>statistics just let you test for the presence or absence of a visible
>bias. As any statistician will tell you, you can't prove the absence
>of an effect by statistical means. You can just prove that it
>didn't show up in your experiment and therefore was less than the
>sensitivity of your test.

Crypto-grade randomness has the negative property that it is
non-deterministic. Statistical tests cannot prove the absence of
determinism in numbers. That is why they cannot be used to
characterize randomness from the numbers themselves.

>Of course, with infinite data, you can develop "tests," in the loosest
>possible sense, of infinite sensitivity.

The sequence 111... with an infinite number of 1s is not a random
number. An infinte random number has no bit bias. Finite random
numbers can have bit bias. Therefore the finite sequnece 111...1 can
be a random number. After all, it is one sequence from a TRNG.

>But this isn't helpful.

It is helpful perhaps when proving theorems. Whether that is of value
to the working cryptanalyst is another matter. Maybe when quantum
computers come online it will be useful.

Bob Knauer

"No Freeman shall ever be debarred the use of arms. The strongest
reason for the people to retain the right to keep and bear arms is,
as a last resort, to protect themselves against tyranny in government."
--Thomas Jefferson

Subject: Re: hardRandNumbGen
Date: Wed, 27 Jan 1999 12:13:29 -0500
From: "Trevor Jackson, III" <fullmoon@aspi.net>
Message-ID: <36AF4939.9C33D7CD@aspi.net>
References: <36af34e2.20943054@nntp.ix.netcom.com>
Newsgroups: sci.crypt
Lines: 15

R. Knauer wrote:

> On 27 Jan 1999 10:21:56 -0500, juola@mathcs.duq.edu (Patrick Juola)
> wrote:
>
> >On the other hand, if I can get a copy of your code, I can just read
> >the code and determine your biases. But you can't rely on keeping
> >your code secret....
>
> I can rely on keeping it just as secret as I keep my keys. If my code
> has been compromised, so have my keys.

Hardly. One can change keys arbitrarily. Once cannot change code so often
(here code == algorithm not .EXE).

Subject: Re: hardRandNumbGen
Date: Wed, 27 Jan 1999 18:44:52 GMT
From: rcktexas@ix.netcom.com (R. Knauer)
Message-ID: <36af5ae4.30673045@nntp.ix.netcom.com>
References: <36AF4939.9C33D7CD@aspi.net>
Newsgroups: sci.crypt
Lines: 31

On Wed, 27 Jan 1999 12:13:29 -0500, "Trevor Jackson, III"
<fullmoon@aspi.net> wrote:

>R. Knauer wrote:

Could I ask you to keep the header information with my attribution,
just like you did for the nested poster below. Thanks.

>> On 27 Jan 1999 10:21:56 -0500, juola@mathcs.duq.edu (Patrick Juola)
>> wrote:

>> >On the other hand, if I can get a copy of your code, I can just read
>> >the code and determine your biases. But you can't rely on keeping
>> >your code secret....

>> I can rely on keeping it just as secret as I keep my keys. If my code
>> has been compromised, so have my keys.

>Hardly. One can change keys arbitrarily. Once cannot change code so often
>(here code == algorithm not .EXE).

How about making your algorithm (code) part of the key? That way you
could change algorithms as often as you change keys.

Bob Knauer

"No Freeman shall ever be debarred the use of arms. The strongest
reason for the people to retain the right to keep and bear arms is,
as a last resort, to protect themselves against tyranny in government."
--Thomas Jefferson

Subject: Re: hardRandNumbGen
Date: 27 Jan 1999 14:32:25 -0500
From: juola@mathcs.duq.edu (Patrick Juola)
Message-ID: <78npk9$m54$1@quine.mathcs.duq.edu>
References: <36af5ae4.30673045@nntp.ix.netcom.com>
Newsgroups: sci.crypt
Lines: 32

In article <36af5ae4.30673045@nntp.ix.netcom.com>,
R. Knauer <rcktexas@ix.netcom.com> wrote:
>On Wed, 27 Jan 1999 12:13:29 -0500, "Trevor Jackson, III"
><fullmoon@aspi.net> wrote:
>
>>R. Knauer wrote:
>
>>> On 27 Jan 1999 10:21:56 -0500, juola@mathcs.duq.edu (Patrick Juola)
>>> wrote:
>
>>> >On the other hand, if I can get a copy of your code, I can just read
>>> >the code and determine your biases. But you can't rely on keeping
>>> >your code secret....
>
>>> I can rely on keeping it just as secret as I keep my keys. If my code
>>> has been compromised, so have my keys.
>
>>Hardly. One can change keys arbitrarily. Once cannot change code so often
>>(here code == algorithm not .EXE).
>
>How about making your algorithm (code) part of the key? That way you
>could change algorithms as often as you change keys.

The larger the key, the more difficulty you have in changing/exchanging
it.

If you're going to exchange 10,000 bytes of code, why don't you use
the same mechanism to exchange a 10,000 byte RSA key or something like
that, which gives you much more bang/buck?

-kitten

Subject: Re: hardRandNumbGen
Date: Wed, 27 Jan 1999 23:46:31 GMT
From: rcktexas@ix.netcom.com (R. Knauer)
Message-ID: <36afa530.49693825@nntp.ix.netcom.com>
References: <78npk9$m54$1@quine.mathcs.duq.edu>
Newsgroups: sci.crypt
Lines: 19

On 27 Jan 1999 14:32:25 -0500, juola@mathcs.duq.edu (Patrick Juola)
wrote:

>The larger the key, the more difficulty you have in changing/exchanging
>it.

>If you're going to exchange 10,000 bytes of code, why don't you use
>the same mechanism to exchange a 10,000 byte RSA key or something like
>that, which gives you much more bang/buck?

The question was theoretical.

Bob Knauer

"No Freeman shall ever be debarred the use of arms. The strongest
reason for the people to retain the right to keep and bear arms is,
as a last resort, to protect themselves against tyranny in government."
--Thomas Jefferson

Subject: Re: hardRandNumbGen
Date: Wed, 27 Jan 1999 15:44:49 -0500
From: "Trevor Jackson, III" <fullmoon@aspi.net>
Message-ID: <36AF7AC1.98B15F0E@aspi.net>
References: <36af5ae4.30673045@nntp.ix.netcom.com>
Newsgroups: sci.crypt
Lines: 33

R. Knauer wrote:

> On Wed, 27 Jan 1999 12:13:29 -0500, "Trevor Jackson, III"
> <fullmoon@aspi.net> wrote:
>

> >R. Knauer wrote:
>
> Could I ask you to keep the header information with my attribution,
> just like you did for the nested poster below. Thanks.
>
> >> On 27 Jan 1999 10:21:56 -0500, juola@mathcs.duq.edu (Patrick Juola)
> >> wrote:
>
> >> >On the other hand, if I can get a copy of your code, I can just read
> >> >the code and determine your biases. But you can't rely on keeping
> >> >your code secret....
>
> >> I can rely on keeping it just as secret as I keep my keys. If my code
> >> has been compromised, so have my keys.
>
> >Hardly. One can change keys arbitrarily. Once cannot change code so often
> >(here code == algorithm not .EXE).
>
> How about making your algorithm (code) part of the key? That way you
> could change algorithms as often as you change keys.

In theory you can do this by encoding the algorithm appropriately. However, you need a clever encoding scheme such that all key values translate to valid algoirhms. In addition, you need to show that all such encoded algorithms are "secure".

Subject: Re: hardRandNumbGen
Date: Wed, 27 Jan 1999 23:50:54 GMT
From: rcktexas@ix.netcom.com (R. Knauer)
Message-ID: <36afa56b.49752720@nntp.ix.netcom.com>
References: <36AF7AC1.98B15F0E@aspi.net>
Newsgroups: sci.crypt
Lines: 29

On Wed, 27 Jan 1999 15:44:49 -0500, "Trevor Jackson, III"
<fullmoon@aspi.net> wrote:

>> How about making your algorithm (code) part of the key? That way you
>> could change algorithms as often as you change keys.

>In theory you can do this by encoding the algorithm appropriately.

Any suggestions on how to do that?

>However, you
>need a clever encoding scheme such that all key values translate to valid
>algoirhms.

A daunting task.

>In addition, you need to show that all such encoded algorithms are
>"secure".

A daunting task.

Bob Knauer

"No Freeman shall ever be debarred the use of arms. The strongest
reason for the people to retain the right to keep and bear arms is,
as a last resort, to protect themselves against tyranny in government."
--Thomas Jefferson

Subject: Re: hardRandNumbGen
Date: 27 Jan 1999 15:06:39 -0500
From: juola@mathcs.duq.edu (Patrick Juola)
Message-ID: <78nrkg$m6o$1@quine.mathcs.duq.edu>
References: <36af34e2.20943054@nntp.ix.netcom.com>
Newsgroups: sci.crypt
Lines: 59

In article <36af34e2.20943054@nntp.ix.netcom.com>,
R. Knauer <rcktexas@ix.netcom.com> wrote:
>>Having only reasonably-sized
>>amounts of funding, computing power, and patience at my disposal,
>>there's a limit to what can be detected -- and if I get something
>>that's indistinguishable from random by this test, then I'm probably
>>willing to pass it as a "good" RNG.
>
>I assumed that you had all the existing resources at your disposal
>that you wanted - like the NSA has.

Um... the NSA does *NOT* have infinite resources. Far from it.
The GNP of the United States is only measured in the trillions,
so any hardware costing more than a quadrillion dollars or so is
probably out of reach.

That's the problem with infinite. It's *BIG*. Of a scale that

dwarfs anything we can possibly imagine. From the standpoint of "perfect secrecy", a block cypher with a 10,000 bit key is exactly as a strong as a Captain Midnight secret decoder ring -- i.e., not perfect and hence infinitely less strong. God has no more difficulty reading message that the NSA would pound against for a billion years than he has reading something rot13ed.

For us mortals, the NSAs capacities are huge. It's easy to hide something from the NSA....

```
>>a complexity curve looking
>>something like this :
>>
>>           _/
>>         _/
>>       _/
>>     _/
>>   _/
>> _/
>>/
>
```

>What is a "complexity curve" and how do you generate one?

x axis == length of string in bits, y axis == linear complexity of the prefix of the (infinite) string of such length. I think there's a brief discussion in the RSA Labs Stream Ciphers tech report -- if not, check the various complexity-related citations contained therein. RSA Labs is on the Web somewhere, as is the TR.

>>The problem is that the Bayesian attack only works if I know -- or
>>can guess -- the type of bias likely to be in your generator.
>
>So, the Bayesian attack is not all that powerful against stream
>ciphers *in general*.

If all you know is that someone used "a stream cypher", then, no, they're not useful. But usually you know more than that. Or unless you guess lucky, which is half the skill in cryptography.

-kitten

Subject: Re: hardRandNumbGen
Date: Wed, 27 Jan 1999 12:10:17 -0500
From: "Trevor Jackson, III" <fullmoon@aspi.net>
Message-ID: <36AF4878.7686E20E@aspi.net>
References: <78nauklks1@quine.mathcs.duq.edu>
Newsgroups: sci.crypt
Lines: 120

Patrick Juola wrote:

> In article <36af1f4b.15416708@nntp.ix.netcom.com>,
> R. Knauer <rcktexas@ix.netcom.com> wrote:
> >On 27 Jan 1999 08:37:36 -0500, juola@mathcs.duq.edu (Patrick Juola)
> >wrote:
> >
> >>You *can't* evaluate a random number generator based only on the
> >>statistical properties of the stream it produces.
> >
> >I fully agree for finite sequences, but as I mention in the thread
> >entitled "Metaphysics of Randomness", statistical tests presumably do
> >apply to infinite sequences. But then, an infinite sequence contains
> >all the details about the generation process, so that makes sense.
>
> Umm.... this is a very limited use of the word "test." I will, of
> course, be delighted to apply any test you choose to any sequence
> you choose, finite or note. But I charge by the hour, and it would,
> of course, be improper of me to give you any results until after I've
> examined the entire sequence.
>
> Still want me to test that infinite sequence for you?
>
> >On an earlier occasion I asked if the Bayesian attack could be used to
> >characterize crypto-grade random numbers. If I gave you a large number
> >of ciphers of substantial length based on some unknown key generation
> >procedure, you tried the Bayesian attack on them and they "leaked" no
> >significant amount of information, then you would probably conclude
> >that the pads were most likely generated by a TRNG, even if I had used
> >some other method like a well-hashed stream cipher.
>
> Probably. By the time you get up to the upper reaches of what we
> can do with PRNGs, the information leak over reasonably-sizes
> samples is pretty infinitesimal. Having only reasonably-sized
> amounts of funding, computing power, and patience at my disposal,
> there's a limit to what can be detected -- and if I get something
> that's indistinguishable from random by this test, then I'm probably
> willing to pass it as a "good" RNG.
>
> On the other hand, there's definitely a possiblity of error there,
> which is why I wouldn't be likely to make any such statement without
> asking to see how you generated the key sequence(s). An example

```
> of something you could "sneak under the radar" would be a very large
> LFSR.  There are well-known tests of linear complexity that give the
> minimum size of an LFSR to generate a given sequence.  A "truly random"
> sequence will (with probability 1) yield a complexity curve looking
> something like this :
>
>            _/
>          _/
>        _/
>      _/
>    _/
>  _/
> /
>
> A LFSR-generated sequence will have a complexity curve something like
> this :
>
>          ____________________________________
>        _/
>      _/
>    _/
>  _/
> /
>
> where it levels off after you hit the generator size.
>
> The question, however, is
>
> a) did you give me enough data to find where the sequence levels off?
> b) did I run a test powerful enough and long enough to find that point?
>
> The first is a mathematical property of the sequence.  The second is
> a question of my time, expertise, and a bit of luck.
>
> The problem is that the Bayesian attack only works if I know -- or
> can guess -- the type of bias likely to be in your generator.  In
> most cases this isn't a problem; many biases are blatantly obvious
> or result from common mistakes.  The more sophisticated you are at
> hiding your biases, the less likely I am to guess the sort of bias
> or to lose patience/funding before I complete the necessary tests.
> (Of course, if I have an infinite amount of time/money, I can
> make an infinite number of guesses and will probably guess right
> on at least one of them.)
>
> On the other hand, if I can get a copy of your code, I can just read
> the code and determine your biases.  But you can't rely on keeping
> your code secret....
>
> > Then I went on to
> >ask if such a "Bayesian Test" for vulnerability could be used to
> >produce confidence limits on the security of subsequent ciphers
> >produced by that encryption method.
>
> In theory, given a large enough  number of computers, without
> question.  But we're running into serious funding difficulties here.
>
> >But why should applying the Bayesian Test to presumably random numbers
> >be any different? (I think I know why - one is a statistical test, the
> >other is a probability survey.)
>
> More a question about finite vs. infinite data.  Bayes' Theorem lets you
> refine hypotheses about biases that you've already made.  Conventional
> statistics just let you test for the presence or absence of a visible
> bias.  As any statistician will tell you, you can't prove the absence
> of an effect by statistical means.  You can just prove that it
> didn't show up in your experiment and therefore was less than the
> sensitivity of your test.
>
> Of course, with infinite data, you can develop "tests," in the loosest
> possible sense, of infinite sensitivity.  But this isn't helpful.

To boil all this verbiage down, analysis of the generation process is more
efficient than analysis of the generator output.  One reason for the
improved efficiency is that understanding the generation process provides
hints of the potential defects worth testing.  Another reason is that the
description of the generation process is logarithmicaly smaller than the
description of the generated output.
```

```
Subject: Re: hardRandNumbGen
Date: Wed, 27 Jan 1999 18:28:35 GMT
From: rcktexas@ix.netcom.com (R. Knauer)
Message-ID: &LT;36af575c.29769466@nntp.ix.netcom.com>
References: &LT;36AF4878.7686E20E@aspi.net>
Newsgroups: sci.crypt
Lines: 40

On Wed, 27 Jan 1999 12:10:17 -0500, "Trevor Jackson, III"
&LT;fullmoon@aspi.net> wrote:

>To boil all this verbiage down, analysis of the generation process is more
>efficient than analysis of the generator output.  One reason for the
>improved efficiency is that understanding the generation process provides
```

>hints of the potential defects worth testing. Another reason is that the
>description of the generation process is logarithmicaly smaller than the
>description of the generated output.

Algorithmic analysis of the generated output itself is completely
worthless if that output is finite in length. If the number 111...1 is
generated by a TRNG, then it is a perfectly valid crypto-grade random
number, just like all the other 2^N-1 numbers of the same length.

Since all of these 2^N sequences of length N are random numbers, what
analysis of them is going to tell you that they are or are not random?
If you apply your analysis to each of the 2^N numbers, then you must
get the same result for each one since they are generated by a TRNG,
otherwise your analysis is fatally flawed.

If your analysis does give the same result for each number, what value
is it in determining randomness or non-randomness? If it correctly
gives the same result for each number, there is no discrimination
being performed. In fact, the analysis needs to say: "I cannot decide
if that number is random or not", for each number, or it is wrong.

If you could prove algorithmically that a given finite sequence is
truly random, then you could also solve both the Godel incompleteness
problem and the Turing halting problem - which are related.

See Chaitin (op. cit.) for the details why.

Bob Knauer

"No Freeman shall ever be debarred the use of arms. The strongest
reason for the people to retain the right to keep and bear arms is,
as a last resort, to protect themselves against tyranny in government."
--Thomas Jefferson

Subject: Re: hardRandNumbGen
Date: Wed, 27 Jan 1999 16:00:55 -0500
From: "Trevor Jackson, III" <fullmoon@aspi.net>
Message-ID: <36AF7E86.637D0856@aspi.net>
References: <36af575c.29769466@nntp.ix.netcom.com>
Newsgroups: sci.crypt
Lines: 67

R. Knauer wrote:

> On Wed, 27 Jan 1999 12:10:17 -0500, "Trevor Jackson, III"
> <fullmoon@aspi.net> wrote:
>
> >To boil all this verbiage down, analysis of the generation process is more
> >efficient than analysis of the generator output. One reason for the
> >improved efficiency is that understanding the generation process provides
> >hints of the potential defects worth testing. Another reason is that the
> >description of the generation process is logarithmicaly smaller than the
> >description of the generated output.
>
> Algorithmic analysis of the generated output itself is completely
> worthless if that output is finite in length. If the number 111...1 is
> generated by a TRNG, then it is a perfectly valid crypto-grade random
> number, just like all the other 2^N-1 numbers of the same length.

False. Analysis of output is insufficient to prove the entropy density is
100%, but it can easily provide useful information. For instance, a page of
business text is around 2^16 bits (of data, not information). If we divide up
the (?)RNG output into samples of this size we can establish a confidence that
the samples are statistically random. If the samples are not statistically
random, we can, with statistical confidence, eliminate that RNG as appropriate
for key generation. If the samples are statistically random we have not proved
that they are appropriate for key generation, but we have shown, to the limit
of our testing capability, the absence of predictability. If the limit of our
testing capability is equal or greater than the capabilities of our
adversaries, then the samples are adequate key material.

Note that the last statement does NOT mean that future samples may be
adequate. The pattern established by the samples inspected might begin
repeating on the very next bit generated. It DOES say that, withint the bounds
of analytic capability, one can inspect keys for effectiveness.

The evaluation of the limits of statistical testing is probably tougher than I
can address because I am insufficiently versed in fundamental statistical
theory. But it's an interesting operational question...

>
>
> Since all of these 2^N sequences of length N are random numbers, what
> analysis of them is going to tell you that they are or are not random?
> If you apply your analysis to each of the 2^N numbers, then you must
> get the same result for each one since they are generated by a TRNG,
> otherwise your analysis is fatally flawed.
>
> If your analysis does give the same result for each number, what value
> is it in determining randomness or non-randomness? If it correctly
> gives the same result for each number, there is no discrimination
> being performed. In fact, the analysis needs to say: "I cannot decide

> if that number is random or not", for each number, or it is wrong.
>
> If you could prove algorithmically that a given finite sequence is
> truly random, then you could also solve both the Godel incompleteness
> problem and the Turing halting problem - which are related.
>
> See Chaitin (op. cit.) for the details why.

Prove? No.

Express confidence in? Certainly.

Arbitrarily high confidence? Yes, given adequate sample size.

Higher confidence than is proper for the other components of the security system? Almost always.

Subject: Re: hardRandNumbGen
Date: 27 Jan 1999 16:28:30 -0500
From: juola@mathcs.duq.edu (Patrick Juola)
Message-ID: <78o0dvmfl1@quine.mathcs.duq.edu>
References: <36AF7E86.637D0856@aspi.net>
Newsgroups: sci.crypt
Lines: 48

In article <36AF7E86.637D0856@aspi.net>,
Trevor Jackson, III <fullmoon@aspi.net> wrote:
>R. Knauer wrote:
>
>> On Wed, 27 Jan 1999 12:10:17 -0500, "Trevor Jackson, III"
>> <fullmoon@aspi.net> wrote:
>>
>> >To boil all this verbiage down, analysis of the generation process is more
>> >efficient than analysis of the generator output. One reason for the
>> >improved efficiency is that understanding the generation process provides
>> >hints of the potential defects worth testing. Another reason is that the
>> >description of the generation process is logarithmicaly smaller than the
>> >description of the generated output.
>>
>> Algorithmic analysis of the generated output itself is completely
>> worthless if that output is finite in length. If the number 111...1 is
>> generated by a TRNG, then it is a perfectly valid crypto-grade random
>> number, just like all the other 2^N-1 numbers of the same length.
>
>False. Analysis of output is insufficient to prove the entropy density is
>100%, but it can easily provide useful information. For instance, a page of
>business text is around 2^16 bits (of data, not information). If we divide up
>the (?)RNG output into samples of this size we can establish a confidence that
>the samples are statistically random. If the samples are not statistically
>random, we can, with statistical confidence, eliminate that RNG as appropriate
>for key generation. If the samples are statistically random we have not proved
>that they are appropriate for key generation, but we have shown, to the limit
>of our testing capability, the absence of predictability. If the limit of our
>testing capability is equal or greater than the capabilities of our
>adversaries, then the samples are adequate key material.
>
>Note that the last statement does NOT mean that future samples may be
>adequate. The pattern established by the samples inspected might begin
>repeating on the very next bit generated. It DOES say that, withint the bounds
>of analytic capability, one can inspect keys for effectiveness.
>
>The evaluation of the limits of statistical testing is probably tougher than I
>can address because I am insufficiently versed in fundamental statistical
>theory. But it's an interesting operational question...

One important operational point should be made here : the distinction
between planned and post-hoc tests. Specifically, if you're going to
reject the RNG on the basis of statistical tests (which is reasonable),
you *NEED* to define the tests you're going to run and the rejection
threshhold before you power up the generator.

-kitten

Subject: Re: hardRandNumbGen
Date: Thu, 28 Jan 1999 18:51:59 -0500
From: "Trevor Jackson, III" <fullmoon@aspi.net>
Message-ID: <36B0F81E.13B14D28@aspi.net>
References: <78o0dvmfl1@quine.mathcs.duq.edu>
Newsgroups: sci.crypt
Lines: 56

Patrick Juola wrote:

> In article <36AF7E86.637D0856@aspi.net>,
> Trevor Jackson, III <fullmoon@aspi.net> wrote:
> >R. Knauer wrote:
> >
> >> On Wed, 27 Jan 1999 12:10:17 -0500, "Trevor Jackson, III"
> >> <fullmoon@aspi.net> wrote:
> >>

> >> >To boil all this verbiage down, analysis of the generation process is more
> >> >efficient than analysis of the generator output. One reason for the
> >> >improved efficiency is that understanding the generation process provides
> >> >hints of the potential defects worth testing. Another reason is that the
> >> >description of the generation process is logarithmicaly smaller than the
> >> >description of the generated output.
> >>
> >> Algorithmic analysis of the generated output itself is completely
> >> worthless if that output is finite in length. If the number 111...1 is
> >> generated by a TRNG, then it is a perfectly valid crypto-grade random
> >> number, just like all the other 2^N-1 numbers of the same length.
> >
> >False. Analysis of output is insufficient to prove the entropy density is
> >100%, but it can easily provide useful information. For instance, a page of
> >business text is around 2^16 bits (of data, not information). If we divide up
> >the (?)RNG output into samples of this size we can establish a confidence that
> >the samples are statistically random. If the samples are not statistically
> >random, we can, with statistical confidence, eliminate that RNG as appropriate
> >for key generation. If the samples are statistically random we have not proved
> >that they are appropriate for key generation, but we have shown, to the limit
> >of our testing capability, the absence of predictability. If the limit of our
> >testing capability is equal or greater than the capabilities of our
> >adversaries, then the samples are adequate key material.
> >
> >Note that the last statement does NOT mean that future samples may be
> >adequate. The pattern established by the samples inspected might begin
> >repeating on the very next bit generated. It DOES say that, withint the bounds
> >of analytic capability, one can inspect keys for effectiveness.
> >
> >The evaluation of the limits of statistical testing is probably tougher than I
> >can address because I am insufficiently versed in fundamental statistical
> >theory. But it's an interesting operational question...
>
> One important operational point should be made here : the distinction
> between planned and post-hoc tests. Specifically, if you're going to
> reject the RNG on the basis of statistical tests (which is reasonable),
> you *NEED* to define the tests you're going to run and the rejection
> threshhold before you power up the generator.

Absolutely. Any other arrangement produces a man-in-the-loop situation. The man isn't the problem, but the loop sure is. The output of the generator should not be able to influence or adjust the acceptance/rejection criteria of a production system.

Of course, during testing, we'll use the test run outputs to monotonicly tighten the criteria until we're ready to go into production.

Subject: Re: hardRandNumbGen
Date: Wed, 27 Jan 1999 23:44:42 GMT
From: rcktexas@ix.netcom.com (R. Knauer)
Message-ID: <36afa3bc.49321650@nntp.ix.netcom.com>
References: <36AF7E86.637D0856@aspi.net>
Newsgroups: sci.crypt
Lines: 38

On Wed, 27 Jan 1999 16:00:55 -0500, "Trevor Jackson, III"
<fullmoon@aspi.net> wrote:

>> Algorithmic analysis of the generated output itself is completely
>> worthless if that output is finite in length. If the number 111...1 is
>> generated by a TRNG, then it is a perfectly valid crypto-grade random
>> number, just like all the other 2^N-1 numbers of the same length.

>False. Analysis of output is insufficient to prove the entropy density is
>100%, but it can easily provide useful information.

I never said it couldn't. However, analysis of the output of a TRNG is
not useful for purposes of crypto.

>For instance, a page of
>business text is around 2^16 bits (of data, not information). If we divide up
>the (?)RNG output into samples of this size we can establish a confidence that
>the samples are statistically random.

Define "statistically random" in terms of finite length numbers.

>If the samples are not statistically
>random, we can, with statistical confidence, eliminate that RNG as appropriate
>for key generation.

Define "not statistically random" in terms of finite length numbers.

Remember we are talking about crypto-grade randomness, where it is
possible to use a random number as a pad in the OTP cryptosystem,
which means it must be proveably secure.

Bob Knauer

"No Freeman shall ever be debarred the use of arms. The strongest
reason for the people to retain the right to keep and bear arms is,
as a last resort, to protect themselves against tyranny in government."
--Thomas Jefferson

Subject: Re: hardRandNumbGen
Date: 28 Jan 1999 11:23:20 -0500
From: juola@mathcs.duq.edu (Patrick Juola)
Message-ID: <78q2to$ng9$1@quine.mathcs.duq.edu>
References: <36afa3bc.49321650@nntp.ix.netcom.com>
Newsgroups: sci.crypt
Lines: 62

In article <36afa3bc.49321650@nntp.ix.netcom.com>,
R. Knauer <rcktexas@ix.netcom.com> wrote:
>On Wed, 27 Jan 1999 16:00:55 -0500, "Trevor Jackson, III"
><fullmoon@aspi.net> wrote:
>>For instance, a page of
>>business text is around 2^16 bits (of data, not information). If we divide up
>>the (?)RNG output into samples of this size we can establish a confidence that
>>the samples are statistically random.
>
>Define "statistically random" in terms of finite length numbers.
>
>>If the samples are not statistically
>>random, we can, with statistical confidence, eliminate that RNG as appropriate
>>for key generation.
>
>Define "not statistically random" in terms of finite length numbers.

In terms of finite length numbers, a finite number is not statistically
random if I can predict it (or something about it).

The key word there is "predict." Post hoc analyses don't mean much.
A priori analyses, however, are a good way of showing that, in practice,
a prediction can be made -- by the simple technique of making and
validating a prediction.

Example.

1) You hand me a black box
2) I tell you 'if this thing doesn't put out "about the same" number
   of ones and zeros, it's hosed and you're fired.'
3) I run the black box for 100,000 bits and it gives 95% ones.
4) I kick your sorry unemployed ass out of the building.

This is valid. *Technically speaking*, there is approximately one
chance in a godzillion that you are not incompetent, but just unlucky.
In practical terms, this wouldn't happen. I'm throwing out one working
RNG every jillion centuries and firing a lot of charletans.

Statistics are like that -- you can never state something "for sure,"
but you can usually make statements like "a truly random system would
only do that 5% of the time, and so I don't think it's random."

The key point above is the ordering of steps 2 and 3. If I reversed
them -- running the generator and *THEN* deciding what counted as
a test of randomness -- then I'd be dealing with a stacked deck.
There's always some test that *can* be run on *any* finite (or
even infinite) sequence that will cause it to appear non-random.
The trick is whether I can *pre*dict that test before examining the
output, or whether I can only hunt for it after the fact.

And, of course, anything that I test for *might* be the result of
a non-random process that "just happened" to give a good result
for the test. I can never prove that a process is "random," but
I can give measures (confidence intervals), stating that the
amount of a particular sort of non-randomness is likely (with
some probability) to be less than some threshhold.

But I can't prove perfection -- I can never get the threshholds
down to zero, which is why I can't prove "randomness", only the
apparent absence of LOTS of randomness.

-kitten

Subject: Re: hardRandNumbGen
Date: Thu, 28 Jan 1999 23:37:14 GMT
From: rcktexas@ix.netcom.com (R. Knauer)
Message-ID: <36b0f08a.10281173@nntp.ix.netcom.com>
References: <78q2to$ng9$1@quine.mathcs.duq.edu>
Newsgroups: sci.crypt
Lines: 49

On 28 Jan 1999 11:23:20 -0500, juola@mathcs.duq.edu (Patrick Juola)
wrote:

>In terms of finite length numbers, a finite number is not statistically
>random if I can predict it (or something about it).

>The key word there is "predict." Post hoc analyses don't mean much.
>A priori analyses, however, are a good way of showing that, in practice,
>a prediction can be made -- by the simple technique of making and
>validating a prediction.

[snip]

Your comments are directed at showing reasons for suspecting that a generator is not random. What do you do if your statistical tests indicate that it IS random?

You throw out the suspect bad generators and argue that such a practice is safe. But what is your argument for the generators that you do not throw out?

I can demonstrate experimentally that a program halts. That's easy. What I cannot demonstrate is that a program does not halt. No matter how long I let it run, there is always the chance that it will halt if I wait just a little longer. If I stop short of that point in time, and conclude that the program does not halt, and yet it would have halted if I had waited just a little longer, then I am in error.

The same is true for your statistical tests. If you had used just a little longer number, you might have found out that the generator was bad. By using numbers of finite length, you are merely guessing that they are random when they pass your statistical tests.

If you could determine that a finite number is truly random by using algorithmic tests, then you could solve the Godel incompleteness problem, the Turing halting problem and the Chaitin complexity problem algorithmically too.

As one poster said earlier, many PRNGs pass statistical tests wonderfully. And cryptanalysts have a real festival when someone uses one of them.

Bob Knauer

"No Freeman shall ever be debarred the use of arms. The strongest reason for the people to retain the right to keep and bear arms is, as a last resort, to protect themselves against tyranny in government."
--Thomas Jefferson

Subject: Re: hardRandNumbGen
Date: 29 Jan 1999 08:59:41 -0500
From: juola@mathcs.duq.edu (Patrick Juola)
Message-ID: <78sesdosg1@quine.mathcs.duq.edu>
References: <36b0f08a.10281173@nntp.ix.netcom.com>
Newsgroups: sci.crypt
Lines: 32

In article <36b0f08a.10281173@nntp.ix.netcom.com>,
R. Knauer <rcktexas@ix.netcom.com> wrote:
>On 28 Jan 1999 11:23:20 -0500, juola@mathcs.duq.edu (Patrick Juola)
>wrote:
>
>>In terms of finite length numbers, a finite number is not statistically
>>random if I can predict it (or something about it).
>
>>The key word there is "predict." Post hoc analyses don't mean much.
>>A priori analyses, however, are a good way of showing that, in practice,
>>a prediction can be made -- by the simple technique of making and
>>validating a prediction.
>
>[snip]
>
>Your comments are directed at showing reasons for suspecting that a
>generator is not random. What do you do if your statistical tests
>indicate that it IS random?
>
>You throw out the suspect bad generators and argue that such a
>practice is safe. But what is your argument for the generators that
>you do not throw out?

That the amount of the bias *that I measured* is less than some threshhold. If I can live with that threshhold and I believe that no other (untested) source of bias is likely to be present, then the cypher is safe to use.

If I don't believe that -- or the threshhold is too high -- then I can't place any reliance on a negative result.

-kitten

Subject: Re: hardRandNumbGen
Date: Sat, 30 Jan 1999 03:18:06 GMT
From: rcktexas@ix.netcom.com (R. Knauer)
Message-ID: <36b279b1.51639443@nntp.ix.netcom.com>
References: <78sesdosg1@quine.mathcs.duq.edu>
Newsgroups: sci.crypt
Lines: 24

On 29 Jan 1999 08:59:41 -0500, juola@mathcs.duq.edu (Patrick Juola)
wrote:

>>You throw out the suspect bad generators and argue that such a

>>practice is safe. But what is your argument for the generators that
>>you do not throw out?

>That the amount of the bias *that I measured* is less than some
>threshhold. If I can live with that threshhold and I believe that
>no other (untested) source of bias is likely to be present, then
>the cypher is safe to use.

>If I don't believe that -- or the threshhold is too high -- then
>I can't place any reliance on a negative result.

Then you accept bad generators and reject good ones.

Bob Knauer

"No Freeman shall ever be debarred the use of arms. The strongest
reason for the people to retain the right to keep and bear arms is,
as a last resort, to protect themselves against tyranny in government."
--Thomas Jefferson

Subject: Re: hardRandNumbGen
Date: 1 Feb 1999 08:55:04 -0500
From: juola@mathcs.duq.edu (Patrick Juola)
Message-ID: <794bnoqdu1@quine.mathcs.duq.edu>
References: <36b279b1.51639443@nntp.ix.netcom.com>
Newsgroups: sci.crypt
Lines: 28

In article <36b279b1.51639443@nntp.ix.netcom.com>,
R. Knauer <rcktexas@ix.netcom.com> wrote:
>On 29 Jan 1999 08:59:41 -0500, juola@mathcs.duq.edu (Patrick Juola)
>wrote:
>
>>>You throw out the suspect bad generators and argue that such a
>>>practice is safe. But what is your argument for the generators that
>>>you do not throw out?
>
>>That the amount of the bias *that I measured* is less than some
>>threshhold. If I can live with that threshhold and I believe that
>>no other (untested) source of bias is likely to be present, then
>>the cypher is safe to use.
>
>>If I don't believe that -- or the threshhold is too high -- then
>>I can't place any reliance on a negative result.
>
>Then you accept bad generators and reject good ones.

Absolutely. With a particular, known, measured probability.

Statisticians call these Type I and Type II errors. The trick is
to make the cost of those errors as small as possible while still
keeping testing costs under control.

-kitten

Subject: Re: hardRandNumbGen
Date: Wed, 03 Feb 1999 18:28:37 GMT
From: rcktexas@ix.netcom.com (R. Knauer)
Message-ID: <36b8755f.10093954@nntp.ix.netcom.com>
References: <794bnoqdu1@quine.mathcs.duq.edu>
Newsgroups: sci.crypt
Lines: 35

On 1 Feb 1999 08:55:04 -0500, juola@mathcs.duq.edu (Patrick Juola)
wrote:

>>Then you accept bad generators and reject good ones.

>Absolutely. With a particular, known, measured probability.

I am not as concerned about rejecting good TRNGs as accepting bad
RNGs, although you aren't gonna do a lot of OTP encryption when your
tests keep rejecting good TRNGs all the time. And your tests will
eventually reject every TRNG you test long enough (I assume correctly
that will sumbit your RNG to tests periodically).

The specification for the OTP does not permit accepting bad RNGs with
any statistical measure that yields a probability other than zero.

>Statisticians call these Type I and Type II errors. The trick is
>to make the cost of those errors as small as possible while still
>keeping testing costs under control.

"Small as possible" is not small enough for the OTP. The probability
for accepting a bad RNG must be zero.

For example, earlier you criticized rejecting the sequence 000...0 for
use as a complete pad on the basis that it would permit the
cryptanalyst to rule out one possible decryption, and yet that string

would certainly be cause for rejecting all TRNGs that produced it,
based on statistical tests.

Bob Knauer

"Sometimes it is said that man cannot be trusted with the government
of himself. Can he, then, be trusted with the government of others?"
--Thomas Jefferson

Subject: Re: hardRandNumbGen
Date: 3 Feb 1999 14:34:40 -0500
From: juola@mathcs.duq.edu (Patrick Juola)
Message-ID: <79a8cg$2up$1@quine.mathcs.duq.edu>
References: <36b8755f.10093954@nntp.ix.netcom.com>
Newsgroups: sci.crypt
Lines: 59

In article <36b8755f.10093954@nntp.ix.netcom.com>,
R. Knauer <rcktexas@ix.netcom.com> wrote:
>On 1 Feb 1999 08:55:04 -0500, juola@mathcs.duq.edu (Patrick Juola)
>wrote:
>
>>>Then you accept bad generators and reject good ones.
>
>>Absolutely. With a particular, known, measured probability.
>
>I am not as concerned about rejecting good TRNGs as accepting bad
>RNGs, although you aren't gonna do a lot of OTP encryption when your
>tests keep rejecting good TRNGs all the time. And your tests will
>eventually reject every TRNG you test long enough (I assume correctly
>that will sumbit your RNG to tests periodically).

Oooooh, repeated testing! Now that's a different kettle of fish.

Actually, I don't need to necessarily reject every TRNG eventually;
I "know" that a given TRNG will fail 1/20 of the tests at a 5%
level, so if I get an approximate failure rate of 1 in 20, then
I keep that one.... and I reject, paradoxically enough, a generator
that doesn't fail 5% of the tests on repeated testing.

And of course, that itself constitutes a meta-test that itself can
be passed or failed with some probability. Eventually along that

But the point is for any well-defined testing scheme, I can make
the probability that a good generator will fail as small as I
like, to the point where the expected number of functioning generators
that I throw out in my professional lifetime is less than 1 (or less
than 1/1000 of a generator, for that matter).

That's not a problem. Yes, *eventually* I will throw out a working
generator. But I'll run out of solar energy to power the testing
equipment at some point, too.

>The specification for the OTP does not permit accepting bad RNGs with
>any statistical measure that yields a probability other than zero.

In that case, you'll never get one. Statistics -- nor engineering
practice, for that matter -- will never allow you to get a probability
all the way down to zero. The best I can do is to get the probability
down to darn-small, where I accept one bad generator per million
lifetimes. There comes a point where keeping a secret becomes silly;
I think the point where you're no longer worried about the NSA and
are now considering Buddha qualifies.

>For example, earlier you criticized rejecting the sequence 000...0 for
>use as a complete pad on the basis that it would permit the
>cryptanalyst to rule out one possible decryption, and yet that string
>would certainly be cause for rejecting all TRNGs that produced it,
>based on statistical tests.

Depends on how often it was produced. Give me a million 20 bit sequences
and I'll bet I can find one. If I find 500,000, I'll worry.

-kitten

Subject: Re: hardRandNumbGen
Date: Fri, 05 Feb 1999 12:31:50 GMT
From: rcktexas@ix.netcom.com (R. Knauer)
Message-ID: <36bae371.2258006@nntp.ix.netcom.com>
References: <79a8cg$2up$1@quine.mathcs.duq.edu>
Newsgroups: sci.crypt
Lines: 26

On 3 Feb 1999 14:34:40 -0500, juola@mathcs.duq.edu (Patrick Juola)
wrote:

>Depends on how often it was produced. Give me a million 20 bit sequences
>and I'll bet I can find one. If I find 500,000, I'll worry.

Let's expand on this a bit, but this time from the point of view of

accepting good RNGs. Imagine that you perform some statistical tests on an RNG and it passes. And you perform some more tests and it passes. Yet the RNG is not crypto-secure. IOW, your tests merely found some statistical properties from which you inferred that the RNG was crypto-grade, like low bias. The number 101010...10 has no bias at all.

So here's the question: Just how much testing do you need to do to convince yourself that a given RNG is crypto-grade (either proveably secure or secure in terms of work effort)?

What happens if you must test forever?

Bob Knauer

"Sometimes it is said that man cannot be trusted with the government of himself. Can he, then, be trusted with the government of others?"
--Thomas Jefferson

Subject: Re: hardRandNumbGen
Date: 4 Feb 1999 14:36:11 -0500
From: hrubin@b.stat.purdue.edu (Herman Rubin)
Message-ID: <79csrb$16je@b.stat.purdue.edu>
References: <794bnoqdu1@quine.mathcs.duq.edu>
Newsgroups: sci.crypt
Lines: 40

In article <794bnoqdu1@quine.mathcs.duq.edu>,
Patrick Juola <juola@mathcs.duq.edu> wrote:
>In article <36b279b1.51639443@nntp.ix.netcom.com>,
>R. Knauer <rcktexas@ix.netcom.com> wrote:
>>On 29 Jan 1999 08:59:41 -0500, juola@mathcs.duq.edu (Patrick Juola)
>>wrote:

>>>>You throw out the suspect bad generators and argue that such a
>>>>practice is safe. But what is your argument for the generators that
>>>>you do not throw out?

>>>That the amount of the bias *that I measured* is less than some
>>>threshhold. If I can live with that threshhold and I believe that
>>>no other (untested) source of bias is likely to be present, then
>>>the cypher is safe to use.

>>>If I don't believe that -- or the threshhold is too high -- then
>>>I can't place any reliance on a negative result.

>>Then you accept bad generators and reject good ones.

>Absolutely. With a particular, known, measured probability.

>Statisticians call these Type I and Type II errors. The trick is
>to make the cost of those errors as small as possible while still
>keeping testing costs under control.

>   -kitten

If one can assume independence between generated sets, one can make
both of these errors quite small, if the point null is not assumed.
That is, the user must specify the inaccuracy allowed, which cannot
be zero.

--
This address is for information only. I do not claim that these views
are those of the Statistics Department or of Purdue University.
Herman Rubin, Dept. of Statistics, Purdue Univ., West Lafayette IN47907-1399
hrubin@stat.purdue.edu   Phone: (765)494-6054   FAX: (765)494-0558

Subject: Re: hardRandNumbGen
Date: Thu, 04 Feb 1999 22:34:11 GMT
From: rcktexas@ix.netcom.com (R. Knauer)
Message-ID: <36ba1b69.44591739@nntp.ix.netcom.com>
References: <79csrb$16je@b.stat.purdue.edu>
Newsgroups: sci.crypt
Lines: 20

On 4 Feb 1999 14:36:11 -0500, hrubin@b.stat.purdue.edu (Herman Rubin)
wrote:

>If one can assume independence between generated sets, one can make
>both of these errors quite small, if the point null is not assumed.
>That is, the user must specify the inaccuracy allowed, which cannot
>be zero.

Please elaborate on how this is done.

I am interested in seeing how much testing is involved for a given
level of confidence, and how that testing effort increases with
decreasing levels of confidence.

Bob Knauer

"Sometimes it is said that man cannot be trusted with the government
of himself. Can he, then, be trusted with the government of others?"
--Thomas Jefferson

Subject: Re: hardRandNumbGen
Date: 5 Feb 1999 08:50:10 -0500
From: juola@mathcs.duq.edu (Patrick Juola)
Message-ID: <79esui$65e$1@quine.mathcs.duq.edu>
References: <36ba1b69.44591739@nntp.ix.netcom.com>
Newsgroups: sci.crypt
Lines: 17

In article <36ba1b69.44591739@nntp.ix.netcom.com>,
R. Knauer <rcktexas@ix.netcom.com> wrote:
>On 4 Feb 1999 14:36:11 -0500, hrubin@b.stat.purdue.edu (Herman Rubin)
>wrote:
>
>>If one can assume independence between generated sets, one can make
>>both of these errors quite small, if the point null is not assumed.
>>That is, the user must specify the inaccuracy allowed, which cannot
>>be zero.
>
>Please elaborate on how this is done.

Well, one starts out by making a questionable assumption, in most cases --
sorry, Dr. Rubin 8-) -- especially when one is testing successive
"pads" from a generator.

-kitten

Subject: Re: hardRandNumbGen
Date: 7 Feb 1999 09:35:58 -0500
From: hrubin@b.stat.purdue.edu (Herman Rubin)
Message-ID: <79k8ce$ms8@b.stat.purdue.edu>
References: <36ba1b69.44591739@nntp.ix.netcom.com>
Newsgroups: sci.crypt
Lines: 43

In article <36ba1b69.44591739@nntp.ix.netcom.com>,
R. Knauer <rcktexas@ix.netcom.com> wrote:
>On 4 Feb 1999 14:36:11 -0500, hrubin@b.stat.purdue.edu (Herman Rubin)
>wrote:

>>If one can assume independence between generated sets, one can make
>>both of these errors quite small, if the point null is not assumed.
>>That is, the user must specify the inaccuracy allowed, which cannot
>>be zero.

>Please elaborate on how this is done.

>I am interested in seeing how much testing is involved for a given
>level of confidence, and how that testing effort increases with
>decreasing levels of confidence.

The number of items to be tested for a given level of confidence
and a given accuracy goes up as the square of the accuracy wanted.
If one had guaranteed independent observations with a probability
p of success, and wanted to get assured probabilities of acceptance
if p=.5, and also of rejection if |p-.5|>d, the sample size needed
would increase as 1/d^2, for d small, which is the situation of
interest here. Not much changes qualitatively if one wants to
accept for p closer to .5 than some multiple of d, such as d/10,
except for a factor.

The same results hold if there are a fixed number of tests; there
would be a factor depending on how many and which tests are to be
used. So a 10 megabyte file could be easily tested in such a way
that one could be fairly sure that the deviation from randomness
was at most 1%, and have good chance of acceptance if it was .1%.

Now if we XOR independent files, say run on different days, the
deviations from randomness multiply. So if we XOR 8 files, and
accept if 5 pass, we can be quite confident about the results.
To do this by testing a single file would require a file of size
more than 10^20.

--
This address is for information only. I do not claim that these views
are those of the Statistics Department or of Purdue University.
Herman Rubin, Dept. of Statistics, Purdue Univ., West Lafayette IN47907-1399
hrubin@stat.purdue.edu Phone: (765)494-6054 FAX: (765)494-0558

Subject: Re: hardRandNumbGen
Date: Sun, 07 Feb 1999 16:29:35 GMT
From: rcktexas@ix.netcom.com (R. Knauer)
Message-ID: <36bdba8f.12639013@nntp.ix.netcom.com>
References: <79k8ce$ms8@b.stat.purdue.edu>
Newsgroups: sci.crypt

Lines: 60

On 7 Feb 1999 09:35:58 -0500, hrubin@b.stat.purdue.edu (Herman Rubin) wrote:

>So a 10 megabyte file could be easily tested in such a way
>that one could be fairly sure that the deviation from randomness
>was at most 1%, and have good chance of acceptance if it was .1%.

I am still having a problem relating this "deviation from randomness" you are testing for and the indeterminancy inherent in a TRNG. You are claiming that a property for infinite numbers applies to finite numbers, albeit with less than probability one.

The thing that really bothers me is that "good chance" part in your statement above. If your tests are probabilistic with only a "good chance" of being correct, then how can they be relied on?

For each test you require a RNG to pass, the builder of the RNG can fake the numbers to pass your tests. Or do you know of a set of tests that can *with absolute certainty* distinguish a faked RNG from a TRNG?

Let me expand on that. It all starts when you come to me and tell me you want a TRNG that will pass your tests. I ask to see your tests so I can test my design for myself before turning the system over to you. Also, I want to see if the tests are reasonable, so I do not waste my time playing amateur games with twit tests.

But unbeknownst to you I am really a purveyor of Snake Oil Generators (SOGs). I take your tests and program an algorithm to generate numbers that will pass your tests, and put that algorithm in a black box so you cannot see it. Even if I dude the black box up with lots of bells ans whistles, I embed the algorithm in the silicon away from your watchful eye. I tell you that the algorithm is used to "diagnose" the SOG so it behaves as certified all the time.

[NB: This is very much like the standard proof of Turing's halting problem.]

Later you come back and complain that the SOG won't pass additional tests that raise your level of confidence, and I point out that you got everything you wanted and paid for. But, since you are a nice gullible customer, I will enhance your SOG for an additional fee, so it will pass your new improved tests. The same scenario obtains until I finally bilk you out of all your money.

You made the most fundamental mistake when you assumed that statistical testing could certify a TRNG to within any arbitrary level of precision. You cannot use deterministic algoritmic tests to certify if numbers are being generated by an indeterministic process. You can only use such tests to certify that the process is not deterministic. Therefore you can not certify that you have a TRNG or a SOG. But that determination is crucial to producing ciphers that are proveably secure.

Bob Knauer

"To compel a man to furnish contributions of money for the propagation of opinions which he disbelieves is sinful and tyrannical."
--Thomas Jefferson

Subject: Re: hardRandNumbGen
Date: 8 Feb 1999 08:49:11 -0500
From: juola@mathcs.duq.edu (Patrick Juola)
Message-ID: <79mq0ntgu1@quine.mathcs.duq.edu>
References: <36bdba8f.12639013@nntp.ix.netcom.com>
Newsgroups: sci.crypt
Lines: 27

In article <36bdba8f.12639013@nntp.ix.netcom.com>,
R. Knauer <rcktexas@ix.netcom.com> wrote:
>On 7 Feb 1999 09:35:58 -0500, hrubin@b.stat.purdue.edu (Herman Rubin)
>wrote:
>
>>So a 10 megabyte file could be easily tested in such a way
>>that one could be fairly sure that the deviation from randomness
>>was at most 1%, and have good chance of acceptance if it was .1%.
>
>I am still having a problem relating this "deviation from randomness"
>you are testing for and the indeterminancy inherent in a TRNG. You are
>claiming that a property for infinite numbers applies to finite
>numbers, albeit with less than probability one.
>
>The thing that really bothers me is that "good chance" part in your
>statement above. If your tests are probabilistic with only a "good
>chance" of being correct, then how can they be relied on?

The "good chance" is quantifiable (as is the "fairly sure").
What is your acceptable degree of uncertainty? What is your
minimal "good chance."

Remember that, according to modern physics, it's only probabilistic
with a good chance that water will get hotter instead of colder
when placed over a lit gas burner.

-kitten

Subject: Re: hardRandNumbGen
Date: Tue, 09 Feb 1999 13:16:14 GMT
From: rcktexas@ix.netcom.com (R. Knauer)
Message-ID: <36c031bb.5684543@nntp.ix.netcom.com>
References: <79mq0ntgu1@quine.mathcs.duq.edu>
Newsgroups: sci.crypt
Lines: 57

On 8 Feb 1999 08:49:11 -0500, juola@mathcs.duq.edu (Patrick Juola)
wrote:

>>The thing that really bothers me is that "good chance" part in your
>>statement above. If your tests are probabilistic with only a "good
>>chance" of being correct, then how can they be relied on?

>The "good chance" is quantifiable (as is the "fairly sure").
>What is your acceptable degree of uncertainty? What is your
>minimal "good chance."

I am not objecting to statistical measure per se. I could not do that
and still believe that Quantum Mechanics is correct.

My objections center on the fact that one cannot characterize the
randomness of a TRNG by measuring the statistical properties of the
numbers it generates. If that were the case, how do you explain what
happens when a PRNG passes such tests, and yet that PRNG is a poor
generator for crypto purposes.

The only way to get a "minimal good chance" of characterizing the TRNG
itself is to do a complete audit on it. That is what experimental
scientists have to do to have a "minimal good chance" of having their
experimental results accepted in the scientific community.

If as a scientist I proposed that the soundness of my experimental
equipment were determined by the output of that equipment, I would be
laughed out of science. Yet that is exactly what is being proposed
here - that the soundness of a TRNG be determined from its output.

>Remember that, according to modern physics, it's only probabilistic
>with a good chance that water will get hotter instead of colder
>when placed over a lit gas burner.

I certainly have no quarrel with statistical mechanics. I once
calculated the probability that all the molecules of air in a room
would end up in the corner. A bit on the small side, so small that it
would never have happened in the age of the Universe.

BTW, I can rig that experiment you describe above to make you think
the phenomenon is happening. I put water in the pot, create a
separation, put in another layer of water and freeze that top layer.
Then I put the thermometer at the bottom, and raise the temperature
just to the point where the ice is about to fall into the bottom. Then
I bring you in to observe how the temperature will decrease when I put
the pot on the stove.

You will only discover what is actually happening if you do a complete
audit of how I prepared the system. The same is true of a TRNG.

Bob Knauer

"The world is filled with violence. Because criminals carry guns,
we decent law-abiding citizens should also have guns. Otherwise
they will win and the decent people will loose."
--James Earl Jones

Subject: Re: hardRandNumbGen
Date: Tue, 09 Feb 1999 11:22:44 -0500
From: "Trevor Jackson, III" <fullmoon@aspi.net>
Message-ID: <36C060D4.CCAAE9E1@aspi.net>
References: <36c031bb.5684543@nntp.ix.netcom.com>
Newsgroups: sci.crypt
Lines: 97

R. Knauer wrote:

> On 8 Feb 1999 08:49:11 -0500, juola@mathcs.duq.edu (Patrick Juola)
> wrote:
>
> >>The thing that really bothers me is that "good chance" part in your
> >>statement above. If your tests are probabilistic with only a "good
> >>chance" of being correct, then how can they be relied on?
>
> >The "good chance" is quantifiable (as is the "fairly sure").
> >What is your acceptable degree of uncertainty? What is your
> >minimal "good chance."

>
> I am not objecting to statistical measure per se. I could not do that
> and still believe that Quantum Mechanics is correct.
>
> My objections center on the fact that one cannot characterize the
> randomness of a TRNG by measuring the statistical properties of the
> numbers it generates. If that were the case, how do you explain what
> happens when a PRNG passes such tests, and yet that PRNG is a poor
> generator for crypto purposes.

How is it that we reach the judgement that an RNG is a poor generator? We may know that it is a deterministic algorithm, and thus weak against certain attacks. These attacks can be considered statistical tests that indicate the strength of the RNG in question.

This argument is ased on the concept that any judgement or decision is going to be based on some evidence gathered by observation and/or experiment. When necessary we can repeat the observation/experiment to classify a phenomenon as subject to the judgement or decision.

Consider the case of a Hardware RNG whose design and implementation appear perfect, but I used too much solder on that connection. I created tiny capacitor, so the inverter somtimes acts as a buffer especially after certain patterns of high frequency data. Statistical testing will indicate that the device is imperfect. The results of the test will be descriptive rather than prescriptive, which may make some people uncomfortable, but it will be crucial to the security of anyone who uses it.

The device owner's adversary is NOT going to simply say, "Oh, he's using an HRNG, so I'll give up now!". The adversary is ging to probe for weaknesses. TEST the cipher system in the fundamental sense of test, which is evaluate. In the case above the adversary is going to find a weakness that leaks information.

Thus our provably secure cipher system turns out not to be. R.I.P.

Any device owner who fails to test for weaknesses is negligent. The term "statistical tests" may conjure up some ivory tower image of irrelevancy or futility, but it actually indicates a skeptical attitude on the part ofa defender of a secret. So the intentions of the tests are appropriate.

The testing technique can be identical to the expected attack. Thus if systems like X are weak because they fall to attack Y, we should utilize attack Y as a test for the weakness demonstrated by system X.

Thus the idean that "bad" RNGs pass statistical tests is a bit of sleight of hand. Reminds me of looking for the keys under the street lamp because the light is better there. Weak RNGs will not pass tests that evaluate cryptographc weakness. RNGs that pass those tests are not weak!

> The only way to get a "minimal good chance" of characterizing the TRNG
> itself is to do a complete audit on it. That is what experimental
> scientists have to do to have a "minimal good chance" of having their
> experimental results accepted in the scientific community.
>
> If as a scientist I proposed that the soundness of my experimental
> equipment were determined by the output of that equipment, I would be
> laughed out of science. Yet that is exactly what is being proposed
> here - that the soundness of a TRNG be determined from its output.
>
> >Remember that, according to modern physics, it's only probabilistic
> >with a good chance that water will get hotter instead of colder
> >when placed over a lit gas burner.
>
> I certainly have no quarrel with statistical mechanics. I once
> calculated the probability that all the molecules of air in a room
> would end up in the corner. A bit on the small side, so small that it
> would never have happened in the age of the Universe.
>
> BTW, I can rig that experiment you describe above to make you think
> the phenomenon is happening. I put water in the pot, create a
> separation, put in another layer of water and freeze that top layer.
> Then I put the thermometer at the bottom, and raise the temperature
> just to the point where the ice is about to fall into the bottom. Then
> I bring you in to observe how the temperature will decrease when I put
> the pot on the stove.
>
> You will only discover what is actually happening if you do a complete
> audit of how I prepared the system. The same is true of a TRNG.

Have you ever audited a system for side effects? Like exhaustive logic tests or provable software design, this kind of analysis is only useful for toy systems.

Subject: Re: hardRandNumbGen
Date: 8 Feb 1999 09:13:28 -0500
From: hrubin@b.stat.purdue.edu (Herman Rubin)
Message-ID: <79mre8$mni@b.stat.purdue.edu>
References: <36bdba8f.12639013@nntp.ix.netcom.com>
Newsgroups: sci.crypt
Lines: 109

In article <36bdba8f.12639013@nntp.ix.netcom.com>,
R. Knauer <rcktexas@ix.netcom.com> wrote:
>On 7 Feb 1999 09:35:58 -0500, hrubin@b.stat.purdue.edu (Herman Rubin)
>wrote:

>>So a 10 megabyte file could be easily tested in such a way
>>that one could be fairly sure that the deviation from randomness
>>was at most 1%, and have good chance of acceptance if it was .1%.

>I am still having a problem relating this "deviation from randomness"
>you are testing for and the indeterminancy inherent in a TRNG. You are
>claiming that a property for infinite numbers applies to finite
>numbers, albeit with less than probability one.

There is a language problem, and it does cause confusion. ANY
physical device producing bits is a TRNG, in the sense that there
is a joint probability distribution of all the bits produced. It
does not follow from this that the probability that any k of the
bits form any particular k-element sequence is exactly 1/2^k.

Your generator, or any other physical generator, produces random
bits. The question remains as to whether they have the particular
properties needed to use these bits as if they had the ideal
properties. It is THIS which is being tested.

>The thing that really bothers me is that "good chance" part in your
>statement above. If your tests are probabilistic with only a "good
>chance" of being correct, then how can they be relied on?

One cannot possibly do better. It is not possible to get certainty,
and one can only do so much to balance the probabilities of the
various kinds of error. This balancing of risks is what theoretical
statisticians study, so that one can decide what can be done and how
well it can be done.

>For each test you require a RNG to pass, the builder of the RNG can
>fake the numbers to pass your tests. Or do you know of a set of tests
>that can *with absolute certainty* distinguish a faked RNG from a
>TRNG?

This is harder than one thinks. There are arguments for PRNGs, among
them being the fact that transmitting them involves much smaller band
width. For cryptography, the criterion is vulnerability of the
transmitted message. For simulation, the criterion is the accuracy
of the calculated results. In neither of these case is the source
of the "random numbers" important, unless it causes the criterion
not to be met. For some simulation problems, quasi random numbers,
which fail independence tests very badly, are known to give better
results than random numbers. We know (in principle) the probability
usually know it for the alternatives.

>Let me expand on that. It all starts when you come to me and tell me
>you want a TRNG that will pass your tests. I ask to see your tests so
>I can test my design for myself before turning the system over to you.
>Also, I want to see if the tests are reasonable, so I do not waste my
>time playing amateur games with twit tests.

>But unbeknownst to you I am really a purveyor of Snake Oil Generators
>(SOGs). I take your tests and program an algorithm to generate numbers
>that will pass your tests, and put that algorithm in a black box so
>you cannot see it. Even if I dude the black box up with lots of bells
>ans whistles, I embed the algorithm in the silicon away from your
>watchful eye. I tell you that the algorithm is used to "diagnose" the
>SOG so it behaves as certified all the time.

As I stated before, this is much harder to do than one would think.
There is a major effort being made to come up with PRNGs which can
meet the "reasonable" tests which have been promulgated. In addition,
these PRNGs are being tried in problems for which some answers can
be computed; a particular generator was found to fail in a simulation
of a physical situation.

Simulations in statistics and physics are using terabytes of PRNGs
now, and will be using thousands or millions of them in the future.

.....

>You made the most fundamental mistake when you assumed that
>statistical testing could certify a TRNG to within any arbitrary level
>of precision. You cannot use deterministic algoritmic tests to certify
>if numbers are being generated by an indeterministic process. You can
>only use such tests to certify that the process is not deterministic.
>Therefore you can not certify that you have a TRNG or a SOG. But that
>determination is crucial to producing ciphers that are proveably
>secure.

What you say is correct. But the type of TRNG you think you have does
not exist, only approximations of it can exist. And it is necessary
to test if the approximations are good enough. A statistical test can
only be algorithmic.

Let us say that my criterion for physical random bits to be good enough
is that the probability of a bit being 1 is .5 within an error of 10^{-10},
given all previous bits. Now how can I test this directly? The answer

is that it cannot be done; if I wanted to test it for all possibilities
of the preceding 100 bits, we already have more than 10^30 cases. It
cannot be done, and so we have to work around it.

Your physical process is not going to be that constant. This is why it
is necessary to make lots of assumptions and test intelligently to hope
to achieve this.

--
This address is for information only. I do not claim that these views
are those of the Statistics Department or of Purdue University.
Herman Rubin, Dept. of Statistics, Purdue Univ., West Lafayette IN47907-1399
hrubin@stat.purdue.edu Phone: (765)494-6054 FAX: (765)494-0558

Subject: Re: hardRandNumbGen
Date: Mon, 08 Feb 1999 15:01:01 GMT
From: rcktexas@ix.netcom.com (R. Knauer)
Message-ID: <36bef38f.9237422@nntp.ix.netcom.com>
References: <79mre8$mni@b.stat.purdue.edu>
Newsgroups: sci.crypt
Lines: 137

On 8 Feb 1999 09:13:28 -0500, hrubin@b.stat.purdue.edu (Herman Rubin)
wrote:

>>I am still having a problem relating this "deviation from randomness"
>>you are testing for and the indeterminancy inherent in a TRNG. You are
>>claiming that a property for infinite numbers applies to finite
>>numbers, albeit with less than probability one.

>There is a language problem, and it does cause confusion.

Indeed!

>ANY physical device producing bits is a TRNG,

Not necessarily the kind of True Random Number Generator that produces
proper sequences for the OTP cryptosystem.

>in the sense that there
>is a joint probability distribution of all the bits produced. It
>does not follow from this that the probability that any k of the
>bits form any particular k-element sequence is exactly 1/2^k.

A TRNG has a specific definition - it must be capable of generating
all possible finite sequences equiprobably. The best example of a TRNG
is a fair coin toss.

>Your generator, or any other physical generator, produces random
>bits.

Not necessarily the kind of random bits that a TRNG produces to
satisfy the requirement above.

>The question remains as to whether they have the particular
>properties needed to use these bits as if they had the ideal
>properties. It is THIS which is being tested.

Here is the crux of the position I am maintaining. I agree that
statistical tests can be used for diagnostic purposes to demonstrate
that a RNG is not a TRNG to within a certain level of confidence. Even
then you can reject a perfectly good TRNG if you do not subject it to
a large number of tests. But that is not where the real danger lies -
the worst is that you will reject properly working TRNGs.

The danger comes about when you attempt to use statistical tests to
demonstrate that a RNG is a crypto-grade TRNG. Even if a RNG passes
your statistical tests, you still do not know if it is a properly
working TRNG or not.

Therefore the very thing you are testing the RNG for, namely its
suitability for use with the OTP system, is not determinable. You
might be able to determine that a RNG is not suitable, but you cannot
determine that an RNG is suitable.

>One cannot possibly do better. It is not possible to get certainty,
>and one can only do so much to balance the probabilities of the
>various kinds of error. This balancing of risks is what theoretical
>statisticians study, so that one can decide what can be done and how
>well it can be done.

You cannot determine with *any* level of confidence that a TRNG is
working properly just because it passes your tests. You can only
determine with a certain level of confidence that it is not working
properly because it does not pass your tests.

>This is harder than one thinks.

I know that.

>There are arguments for PRNGs, among
>them being the fact that transmitting them involves much smaller band

>width. For cryptography, the criterion is vulnerability of the
>transmitted message. For simulation, the criterion is the accuracy
>of the calculated results. In neither of these case is the source
>of the "random numbers" important, unless it causes the criterion
>not to be met.

I disagree with your statement for purposes of the OTP cryptosystem. The security of the OTP relies on the fact that the pad is constructed from the output of a TRNG. If you use a source of "randomness" that does not meet the specification of the TRNG, you could suffer a probabilistic attack.

Only if the numbers come from a completely nondeterministic random number generator can there be no "leakage" of information.

>>You made the most fundamental mistake when you assumed that
>>statistical testing could certify a TRNG to within any arbitrary level
>>of precision. You cannot use deterministic algoritmic tests to certify
>>if numbers are being generated by an indeterministic process. You can
>>only use such tests to certify that the process is not deterministic.
>>Therefore you can not certify that you have a TRNG or a SOG. But that
>>determination is crucial to producing ciphers that are proveably
>>secure.

>What you say is correct.

Finally! I got someone to agree with me on something!

>But the type of TRNG you think you have does
>not exist, only approximations of it can exist. And it is necessary
>to test if the approximations are good enough.

You can never know if they are good enough, only that they are bad enough.

I realize that an actual TRNG will not be perfect, but I believe it can be made to specification such that it will not cause any significant amount of information to "leak".

That may also be the case for some stream ciphers, such as using the least significant bits of text and hashing them to high entropy density. Even though the latter is not a nondeterministic TRNG, it may not leak cause enough information to leak for practical purposes.

My concern is that statistical tests cannot be used to determine whether ANY method of random number generation is suitable for the OTP system. All it can be used for is to reject those RNGs that do not pass, and then only probabilistically. The only way you can certify a TRNG for use with the OTP system is to do a full audit of the generation method.

>Your physical process is not going to be that constant. This is why it
>is necessary to make lots of assumptions and test intelligently to hope
>to achieve this.

I believe it is possible to build a TRNG that can be be certified to be properly working based on subsystem diagnostics and the knowledge of the random process that gives it its nondeterminancy - without having to resort to statistical tests on the output sequences.

Bob Knauer

"The world is filled with violence. Because criminals carry guns, we decent law-abiding citizens should also have guns. Otherwise they will win and the decent people will loose."
--James Earl Jones

Subject: Re: hardRandNumbGen
Date: 8 Feb 1999 10:42:38 -0500
From: juola@mathcs.duq.edu (Patrick Juola)
Message-ID: <79n0le$u0h$1@quine.mathcs.duq.edu>
References: <36bef38f.9237422@nntp.ix.netcom.com>
Newsgroups: sci.crypt
Lines: 73

In article <36bef38f.9237422@nntp.ix.netcom.com>,
R. Knauer <rcktexas@ix.netcom.com> wrote:
>On 8 Feb 1999 09:13:28 -0500, hrubin@b.stat.purdue.edu (Herman Rubin)
>wrote:
>
>>>I am still having a problem relating this "deviation from randomness"
>>>you are testing for and the indeterminancy inherent in a TRNG. You are
>>>claiming that a property for infinite numbers applies to finite
>>>numbers, albeit with less than probability one.
>
>>There is a language problem, and it does cause confusion.
>
>Indeed!
>
>>ANY physical device producing bits is a TRNG,
>
>Not necessarily the kind of True Random Number Generator that produces

>proper sequences for the OTP cryptosystem.
>
>>in the sense that there
>>is a joint probability distribution of all the bits produced. It
>>does not follow from this that the probability that any k of the
>>bits form any particular k-element sequence is exactly 1/2^k.
>
>A TRNG has a specific definition - it must be capable of generating
>all possible finite sequences equiprobably. The best example of a TRNG
>is a fair coin toss.
>
>>Your generator, or any other physical generator, produces random
>>bits.
>
>Not necessarily the kind of random bits that a TRNG produces to
>satisfy the requirement above.
>
>>The question remains as to whether they have the particular
>>properties needed to use these bits as if they had the ideal
>>properties. It is THIS which is being tested.
>
>Here is the crux of the position I am maintaining. I agree that
>statistical tests can be used for diagnostic purposes to demonstrate
>that a RNG is not a TRNG to within a certain level of confidence. Even
>then you can reject a perfectly good TRNG if you do not subject it to
>a large number of tests. But that is not where the real danger lies -
>the worst is that you will reject properly working TRNGs.
>
>The danger comes about when you attempt to use statistical tests to
>demonstrate that a RNG is a crypto-grade TRNG. Even if a RNG passes
>your statistical tests, you still do not know if it is a properly
>working TRNG or not.
>
>Therefore the very thing you are testing the RNG for, namely its
>suitability for use with the OTP system, is not determinable. You
>might be able to determine that a RNG is not suitable, but you cannot
>determine that an RNG is suitable.

No. There are two things you need to do to produce a certifiable
TRNG.

One is to confirm that the device is, in fact, a "random number generator"
in the sense that it produces random bits. The main thing to confirm
then is that you can get an unbounded number of random (although not
necessarily equiprobable) bits out of the system. This requires
examination of the generator -- and is probably impossible unless
you're willing to make certain assumptions about various physical
processes such as radioactive decay or wave height or something.

The other is to confirm that the outputs are bias-free -- or more
accuratley as bias-free as possible, since there's no way to prove
ZERO bias. And this is best done statistically, although if you
really trust your engineers you can probably do it by design analysis
as well.

-kitten

Subject: Re: hardRandNumbGen
Date: Tue, 09 Feb 1999 18:34:07 GMT
From: rcktexas@ix.netcom.com (R. Knauer)
Message-ID: <36c07adb.992577@nntp.ix.netcom.com>
References: <79n0le$u0h$1@quine.mathcs.duq.edu>
Newsgroups: sci.crypt
Lines: 78

On 8 Feb 1999 10:42:38 -0500, juola@mathcs.duq.edu (Patrick Juola)
wrote:

>>Therefore the very thing you are testing the RNG for, namely its
>>suitability for use with the OTP system, is not determinable. You
>>might be able to determine that a RNG is not suitable, but you cannot
>>determine that an RNG is suitable.

>No. There are two things you need to do to produce a certifiable
>TRNG.

I meant "you cannot determine that an RNG is suitable"... using
statistical tests on the output.

>One is to confirm that the device is, in fact, a "random number generator"
>in the sense that it produces random bits. The main thing to confirm
>then is that you can get an unbounded number of random (although not
>necessarily equiprobable) bits out of the system.

I do not know what you mean by "random" in that sentence. I will take
it to mean "indeterminant".

Which brings up a question I was going to bring up earlier and have
been waiting for the right place. We speak of the ills of bit-bias in
terms of random number generation, but what if the generator were
designed with a deliberate bias? As an analog (and only as an analog)
imagine a symmetric polygonal die with one more 1 than 0. That would
have a built in bias, yet each outcome of a throw would be

indeterminant. So you subject the output of that die to a statistical test for bit-bias and it flunks. Now what?

Also, imagine actually using the output for an OTP and your attacker tries to figure out why the bits in the ciphers are biased. Will that do him any good? IOW, does using the pad from a deliberately biased RNG (which is otherwise completely indeterminant) leak any information that is useful for decrypting your ciphers?

It would seem that any bias, even bias that is deliberately introduced and accounted for, is going to weaken the random number generation process cryptographically, since in the limit that the bias becomes very large, you have a totally unsecure system? Yet the TNG is completely indeterminant from one throw of the die to the next

[NB: For those of you who were here a year ago, this very important point was discussed at length - and is the reason we define a TRNG in terms of equiprobable sequences, and not just independent bit generation.]

>This requires
>examination of the generator -- and is probably impossible unless
>you're willing to make certain assumptions about various physical
>processes such as radioactive decay or wave height or something.

Therefore you must have a known source of randomness to avoid such assumptions. Radioactive decay suffices - unless you are prepared to take on the entire scientific community with a refutation of indeterminancy in Quantum Mechanics, in which case be sure to bring your lunch because you are gonna be at it for a while.

>The other is to confirm that the outputs are bias-free -- or more
>accuratley as bias-free as possible, since there's no way to prove
>ZERO bias. And this is best done statistically, although if you
>really trust your engineers you can probably do it by design analysis
>as well.

If you know that your RNG is supposed to be bias-free, then testing it for bias may be necessary but is certainly not sufficient to demonstrate that it is working properly - with the proviso that you know that it is designed to be a TRNG so you can avoid the possibility that you have a PRNG which passes the tests and fools you.

Bob Knauer

"The world is filled with violence. Because criminals carry guns, we decent law-abiding citizens should also have guns. Otherwise they will win and the decent people will loose."
--James Earl Jones

Subject: Re: hardRandNumbGen
Date: Wed, 10 Feb 1999 00:05:49 -0500
From: "Trevor Jackson, III" <fullmoon@aspi.net>
Message-ID: <36C113AC.E21DD743@aspi.net>
References: <36c07adb.992577@nntp.ix.netcom.com>
Newsgroups: sci.crypt
Lines: 87

R. Knauer wrote:

> On 8 Feb 1999 10:42:38 -0500, juola@mathcs.duq.edu (Patrick Juola)
> wrote:
>
> >>Therefore the very thing you are testing the RNG for, namely its
> >>suitability for use with the OTP system, is not determinable. You
> >>might be able to determine that a RNG is not suitable, but you cannot
> >>determine that an RNG is suitable.
>
> >No. There are two things you need to do to produce a certifiable
> >TRNG.
>
> I meant "you cannot determine that an RNG is suitable"... using
> statistical tests on the output.
>
> >One is to confirm that the device is, in fact, a "random number generator"
> >in the sense that it produces random bits. The main thing to confirm
> >then is that you can get an unbounded number of random (although not
> >necessarily equiprobable) bits out of the system.
>
> I do not know what you mean by "random" in that sentence. I will take
> it to mean "indeterminant".
>
> Which brings up a question I was going to bring up earlier and have
> been waiting for the right place. We speak of the ills of bit-bias in
> terms of random number generation, but what if the generator were
> designed with a deliberate bias? As an analog (and only as an analog)
> imagine a symmetric polygonal die with one more 1 than 0. That would
> have a built in bias, yet each outcome of a throw would be
> indeterminant. So you subject the output of that die to a statistical
> test for bit-bias and it flunks. Now what?
>
> Also, imagine actually using the output for an OTP and your attacker

> tries to figure out why the bits in the ciphers are biased. Will that
> do him any good? IOW, does using the pad from a deliberately biased
> RNG (which is otherwise completely indeterminant) leak any information
> that is useful for decrypting your ciphers?
>
> It would seem that any bias, even bias that is deliberately introduced
> and accounted for, is going to weaken the random number generation
> process cryptographically, since in the limit that the bias becomes
> very large, you have a totally unsecure system? Yet the TNG is
> completely indeterminant from one throw of the die to the next
>
> [NB: For those of you who were here a year ago, this very important
> point was discussed at length - and is the reason we define a TRNG in
> terms of equiprobable sequences, and not just independent bit
> generation.]
>
> >This requires
> >examination of the generator -- and is probably impossible unless
> >you're willing to make certain assumptions about various physical
> >processes such as radioactive decay or wave height or something.
>
> Therefore you must have a known source of randomness to avoid such
> assumptions. Radioactive decay suffices - unless you are prepared to
> take on the entire scientific community with a refutation of
> indeterminancy in Quantum Mechanics, in which case be sure to bring
> your lunch because you are gonna be at it for a while.

Any tests you would use to prove QM indeterminate can be used t prove a non QM
RNG indeterminate. naturally, these would be statistical tests.

>
>
> >The other is to confirm that the outputs are bias-free -- or more
> >accuratley as bias-free as possible, since there's no way to prove
> >ZERO bias. And this is best done statistically, although if you
> >really trust your engineers you can probably do it by design analysis
> >as well.
>
> If you know that your RNG is supposed to be bias-free, then testing it
> for bias may be necessary but is certainly not sufficient to
> demonstrate that it is working properly - with the proviso that you
> know that it is designed to be a TRNG so you can avoid the possibility
> that you have a PRNG which passes the tests and fools you.
>
> Bob Knauer
>
> "The world is filled with violence. Because criminals carry guns,
> we decent law-abiding citizens should also have guns. Otherwise
> they will win and the decent people will loose."
> --James Earl Jones

Subject: Re: hardRandNumbGen
Date: Wed, 10 Feb 1999 15:00:04 GMT
From: rcktexas@ix.netcom.com (R. Knauer)
Message-ID: <36c19eb3.4946452@nntp.ix.netcom.com>
References: <36C113AC.E21DD743@aspi.net>
Newsgroups: sci.crypt
Lines: 15

On Wed, 10 Feb 1999 00:05:49 -0500, "Trevor Jackson, III"
<fullmoon@aspi.net> wrote:

>Any tests you would use to prove QM indeterminate can be used t prove a non QM
>RNG indeterminate. naturally, these would be statistical tests.

Apparently you never mastered Quantum Mechanics, or you would not be
making such a ludicrous statement as you did above.

Bob Knauer

"It is not a matter of what is true that counts, but a matter of
what is perceived to be true."
--Henry Kissinger

Subject: Re: hardRandNumbGen
Date: 10 Feb 1999 08:53:19 -0500
From: juola@mathcs.duq.edu (Patrick Juola)
Message-ID: <79s30f$9db$1@quine.mathcs.duq.edu>
References: <36c07adb.992577@nntp.ix.netcom.com>
Newsgroups: sci.crypt
Lines: 64

In article <36c07adb.992577@nntp.ix.netcom.com>,
R. Knauer <rcktexas@ix.netcom.com> wrote:
>On 8 Feb 1999 10:42:38 -0500, juola@mathcs.duq.edu (Patrick Juola)
>wrote:
>
>>>Therefore the very thing you are testing the RNG for, namely its

>>>suitability for use with the OTP system, is not determinable. You
>>>might be able to determine that a RNG is not suitable, but you cannot
>>>determine that an RNG is suitable.
>
>>No. There are two things you need to do to produce a certifiable
>>TRNG.
>
>I meant "you cannot determine that an RNG is suitable"... using
>statistical tests on the output.
>
>>One is to confirm that the device is, in fact, a "random number generator"
>>in the sense that it produces random bits. The main thing to confirm
>>then is that you can get an unbounded number of random (although not
>>necessarily equiprobable) bits out of the system.
>
>I do not know what you mean by "random" in that sentence. I will take
>it to mean "indeterminant".
>
>Which brings up a question I was going to bring up earlier and have
>been waiting for the right place. We speak of the ills of bit-bias in
>terms of random number generation, but what if the generator were
>designed with a deliberate bias? As an analog (and only as an analog)
>imagine a symmetric polygonal die with one more 1 than 0. That would
>have a built in bias, yet each outcome of a throw would be
>indeterminant. So you subject the output of that die to a statistical
>test for bit-bias and it flunks. Now what?

Depends. If it's a really good generator with a known bias, there
are mathematical techniques that will allow me to strip out the bias
and produced an unbiased stream. So if I'm willing to embed the
hardware in some other equipment, it may be a good building block.

>Also, imagine actually using the output for an OTP and your attacker
>tries to figure out why the bits in the ciphers are biased. Will that
>do him any good? IOW, does using the pad from a deliberately biased
>RNG (which is otherwise completely indeterminant) leak any information
>that is useful for decrypting your ciphers?

Broadly speaking, yes. First, I can start decrypting at the most
probable key and work downwards from there. Second, I can produce
a probability (Bayes' theorem again) for every potential key, and
from that derive probabilities for various plaintexts. If you
send a message telling your broker either to "buy!" or "sell", but
one is overwhelmingly more probable than the other, that's a
potentially serious crack.

>It would seem that any bias, even bias that is deliberately introduced
>and accounted for, is going to weaken the random number generation
>process cryptographically, since in the limit that the bias becomes
>very large, you have a totally unsecure system? Yet the TNG is
>completely indeterminant from one throw of the die to the next

But you can strip out bias fairly easily -- gather bits in pairs, output
a 1 bit if the pair is 01, a zero bit if the pair is 10 and gather a
new pair otherwise. Yes, you waste about 75 percent of your generator
this way.... but bits are cheap.

-kitten

Subject: Re: hardRandNumbGen
Date: Thu, 11 Feb 1999 08:56:20 -0700
From: "Tony T. Warnock" <u091889@cic-mail.lanl.gov>
Message-ID: <36C2FDA4.5B0776F7@cic-mail.lanl.gov>
References: <79s30f$9db$1@quine.mathcs.duq.edu>
Newsgroups: sci.crypt
Lines: 11

Patrick Juola wrote:

> But you can strip out bias fairly easily -- gather bits in pairs, output
> a 1 bit if the pair is 01, a zero bit if the pair is 10 and gather a
> new pair otherwise. Yes, you waste about 75 percent of your generator
> this way.... but bits are cheap.

Actually, "random" bits are quite expensive to get. "Ignorance can be costly."

Tony

Subject: Re: hardRandNumbGen
Date: Thu, 11 Feb 1999 18:39:51 GMT
From: rcktexas@ix.netcom.com (R. Knauer)
Message-ID: <36c31cb8.17085527@nntp.ix.netcom.com>
References: <79s30f$9db$1@quine.mathcs.duq.edu>
Newsgroups: sci.crypt
Lines: 48

On 10 Feb 1999 08:53:19 -0500, juola@mathcs.duq.edu (Patrick Juola)
wrote:

>Depends. If it's a really good generator with a known bias, there
>are mathematical techniques that will allow me to strip out the bias

>and produced an unbiased stream.

This brings up the question of whether anti-skewing changes the equiprobability of the generator. I suspect it does for the following reason.

A TRNG must be capable of generating all possible finite sequences equiprobably. If it is biased, then is it not doing that. Anti-skewing procedures do not generate those sequences that under-representted because of the bias.

>>IOW, does using the pad from a deliberately biased
>>RNG (which is otherwise completely indeterminant) leak any information
>>that is useful for decrypting your ciphers?

>Broadly speaking, yes.

Just as I suspected - equiprobability is at the heart of proveably secure crypto. Equiprobability implies both independence and equidistribution of all possible finite sequences. Independence is not sufficient and equidistribution is not sufficient - you must have both to satisfy the requirements for proveable security.

>But you can strip out bias fairly easily -- gather bits in pairs, output
>a 1 bit if the pair is 01, a zero bit if the pair is 10 and gather a
>new pair otherwise. Yes, you waste about 75 percent of your generator
>this way.... but bits are cheap.

See my comments above. I believe that if you employ that, or any other anti-skewing technique, to the output of a TRNG which is suffering from severe bias, then you will no longer have a TRNG in the sense of its fundamental specification - capable of generating all possible finite sequences equiprobably.

IOW, I fail to see how any anti-skewing procedure, including the one you gave above, is going to resore the equiprobability of the TRNG.

Bob Knauer

"It is not a matter of what is true that counts, but a matter of what is perceived to be true."
--Henry Kissinger

Subject: Re: hardRandNumbGen
Date: 11 Feb 1999 15:42:48 -0500
From: juola@mathcs.duq.edu (Patrick Juola)
Message-ID: <79vfc8$d9q$1@quine.mathcs.duq.edu>
References: <36c31cb8.17085527@nntp.ix.netcom.com>
Newsgroups: sci.crypt
Lines: 36

In article <36c31cb8.17085527@nntp.ix.netcom.com>,
R. Knauer <rcktexas@ix.netcom.com> wrote:
>On 10 Feb 1999 08:53:19 -0500, juola@mathcs.duq.edu (Patrick Juola)
>wrote:
>
>>Depends. If it's a really good generator with a known bias, there
>>are mathematical techniques that will allow me to strip out the bias
>>and produced an unbiased stream.
>
>This brings up the question of whether anti-skewing changes the
>equiprobability of the generator. I suspect it does for the following
>reason.
>
>A TRNG must be capable of generating all possible finite sequences
>equiprobably. If it is biased, then is it not doing that. Anti-skewing
>procedures do not generate those sequences that under-representted
>because of the bias.

Um... This is simply incorrect. That's *exactly* what anti-skewing procedures do.

Think of it this way. Assume you have a biased, but independent, bit source that generates 1s with probability p > 0.5. Consider two successive bits, x, y.

The probability of getting the sequence 1, 0 is p * (1-p).
The probability of getting the sequence 0, 1 is (1-p) * p, which is identical to the above.

So if you output a 1 bit when you see the pair 1,0 and a zero bit when you see the pair 0,1 (and nothing otherwise), then you've got a provably unbiased output stream -- the bias has been scrubbed from the input -- by the technique of throwing away everyting that *isn't* unbiased, broadly speaking.

-kitten

Subject: Re: hardRandNumbGen
Date: Sat, 13 Feb 1999 15:31:48 GMT
From: rcktexas@ix.netcom.com (R. Knauer)

Message-ID: <36c598cd.7094651@nntp.ix.netcom.com>
References: <79vfc8$d9q$1@quine.mathcs.duq.edu>
Newsgroups: sci.crypt
Lines: 49

On 11 Feb 1999 15:42:48 -0500, juola@mathcs.duq.edu (Patrick Juola)
wrote:

>>A TRNG must be capable of generating all possible finite sequences
>>equiprobably. If it is biased, then is it not doing that. Anti-skewing
>>procedures do not generate those sequences that under-representted
>>because of the bias.

>Um... This is simply incorrect. That's *exactly* what anti-skewing
>procedures do.

I would like to see how that is possible.

>Think of it this way. Assume you have a biased, but independent, bit
>source that generates 1s with probability p > 0.5. Consider two
>successive bits, x, y.

>The probability of getting the sequence 1, 0 is p * (1-p).
>The probability of getting the sequence 0, 1 is (1-p) * p, which is
>identical to the above.

>So if you output a 1 bit when you see the pair 1,0 and a zero bit
>when you see the pair 0,1 (and nothing otherwise), then you've
>got a provably unbiased output stream -- the bias has been scrubbed
>from the input -- by the technique of throwing away everyting that
>*isn't* unbiased, broadly speaking.

I believe that algorithm is attributed to Knuth.

This all sounds good on the surface but I am not convinced at this
point. For example, there are sequences from a TRNG that are heavily
biased, like 111...1 and 000...0, yet the scheme above does not seem
to be able to produce those sequences. Can that anti-skewing technique
above generate all possible sequences including 111...1 and 000...0
with their expected probability based on their length? Is so, please
show us how.

If there is a reference in Li & Vitanyi, please cite it, since I have
renewed the book for a couple more weeks for a second reading. If not,
perhaps you can provide another reference. I might as well get as much
out of my "free" library as I can, since I am paying for it.

Bob Knauer

"The one thing every man fears is the unknown. When presented with this
scenario, individual rights will be willingly relinquished for the guarantee
of their well being granted to them by their world government."
--Henry Kissinger

Subject: Re: hardRandNumbGen
Date: Sat, 13 Feb 1999 09:28:03 -0800
From: "karl malbrain" <karl_m@acm.org>
Message-ID: <rDix2.16$Gr2.23@typhoon-la.pbi.net>
References: <36c598cd.7094651@nntp.ix.netcom.com>
Newsgroups: sci.crypt
Lines: 28

R. Knauer <rcktexas@ix.netcom.com> wrote in message
news:36c598cd.7094651@nntp.ix.netcom.com...
>On 11 Feb 1999 15:42:48 -0500, juola@mathcs.duq.edu (Patrick Juola)
>wrote:
>(...)
>>Think of it this way. Assume you have a biased, but independent, bit
>>source that generates 1s with probability p > 0.5. Consider two
>>successive bits, x, y.
>
>>The probability of getting the sequence 1, 0 is p * (1-p).
>>The probability of getting the sequence 0, 1 is (1-p) * p, which is
>>identical to the above.
>
>>So if you output a 1 bit when you see the pair 1,0 and a zero bit
>>when you see the pair 0,1 (and nothing otherwise), then you've
>>got a provably unbiased output stream -- the bias has been scrubbed
>>from the input -- by the technique of throwing away everyting that
>>*isn't* unbiased, broadly speaking.
>
>I believe that algorithm is attributed to Knuth.
>
What you're missing here is INDUCTION. The pair-wise bias of the
generator's bits is being extended by you to define a group-wise bias (take
this in the negative). Karl M

Subject: Re: hardRandNumbGen
Date: 15 Feb 1999 09:18:14 -0500
From: juola@mathcs.duq.edu (Patrick Juola)
Message-ID: <7a9ab6dio1@quine.mathcs.duq.edu>
References: <36c598cd.7094651@nntp.ix.netcom.com>
Newsgroups: sci.crypt
Lines: 73

In article <36c598cd.7094651@nntp.ix.netcom.com>,
R. Knauer <rcktexas@ix.netcom.com> wrote:
>On 11 Feb 1999 15:42:48 -0500, juola@mathcs.duq.edu (Patrick Juola)
>wrote:
>
>>>A TRNG must be capable of generating all possible finite sequences
>>>equiprobably. If it is biased, then is it not doing that. Anti-skewing
>>>procedures do not generate those sequences that under-representted
>>>because of the bias.
>
>>Um... This is simply incorrect. That's *exactly* what anti-skewing
>>procedures do.
>
>I would like to see how that is possible.
>
>>Think of it this way. Assume you have a biased, but independent, bit
>>source that generates 1s with probability p > 0.5. Consider two
>>successive bits, x, y.
>
>>The probability of getting the sequence 1, 0 is p * (1-p).
>>The probability of getting the sequence 0, 1 is (1-p) * p, which is
>>identical to the above.
>
>>So if you output a 1 bit when you see the pair 1,0 and a zero bit
>>when you see the pair 0,1 (and nothing otherwise), then you've
>>got a provably unbiased output stream -- the bias has been scrubbed
>>from the input -- by the technique of throwing away everyting that
>>*isn't* unbiased, broadly speaking.
>
>I believe that algorithm is attributed to Knuth.

Before that, it was attributed to Von Neumann. it's an old one.

>This all sounds good on the surface but I am not convinced at this
>point. For example, there are sequences from a TRNG that are heavily
>biased, like 111...1 and 000...0, yet the scheme above does not seem
>to be able to produce those sequences. Can that anti-skewing technique
>above generate all possible sequences including 111...1 and 000...0
>with their expected probability based on their length? Is so, please
>show us how.

Certainly. Remember that the symbol (1) is generated from a subsequence
of 10 (or 1110, 0010, etc... in fact, by any sequence of the form
[(00)+(11)*10]. So any sequence of the form [(00)+(11)*10]*
will "debiasify" into 11111...11...

As to proof that the expected probability is based on the length;
do it by case analysis and induction.

There are four cases for the first four bits; as expressed above :

>>The probability of getting the sequence 1, 0 is p * (1-p).
>>The probability of getting the sequence 0, 1 is (1-p) * p.

  The probability of getting the sequence 1, 1 is p*p.
  The probability of getting the sequence 0, 0 is (1-p)*(1-p).

Note : only the first two generate output.

Assume for induction that all (output) sequences of length K or less
are printed uniformly with probability proportional to their length.
(The base case of length zero sequences is, literally, trivial).
Assume for contradiction, wolog, that the sequence X1 is more
probable than the sequence X0. This means that it's more likely
that a 1 was printed than a 0. But *this*, in turn, means that
the sequence [(00)+(11)*10] was *generated* more probably than
the sequence [(00)+(11)*01]. The prefix probability is the same
in both cases, so the only possible source of such difference is
in the probability of generating 10 vs. 01. But by case analysis,
these probabilities are equal, hence the desired contraction is
obtained.

        -kitten

Subject: Re: hardRandNumbGen
Date: Mon, 15 Feb 1999 12:10:49 -0500
From: "Trevor Jackson, III" <fullmoon@aspi.net>
Message-ID: <36C85518.D62EF8@aspi.net>
References: <7a9ab6dio1@quine.mathcs.duq.edu>
Newsgroups: sci.crypt
Lines: 88

Patrick Juola wrote:

> In article <36c598cd.7094651@nntp.ix.netcom.com>,
> R. Knauer <rcktexas@ix.netcom.com> wrote:
> >On 11 Feb 1999 15:42:48 -0500, juola@mathcs.duq.edu (Patrick Juola)
> >wrote:
> >
> >>>A TRNG must be capable of generating all possible finite sequences
> >>>equiprobably. If it is biased, then is it not doing that. Anti-skewing
> >>>procedures do not generate those sequences that under-representted
> >>>because of the bias.
> >
> >>Um... This is simply incorrect. That's *exactly* what anti-skewing
> >>procedures do.
> >
> >I would like to see how that is possible.
> >
> >>Think of it this way. Assume you have a biased, but independent, bit
> >>source that generates 1s with probability p > 0.5. Consider two
> >>successive bits, x, y.
> >
> >>The probability of getting the sequence 1, 0 is p * (1-p).
> >>The probability of getting the sequence 0, 1 is (1-p) * p, which is
> >>identical to the above.
> >
> >>So if you output a 1 bit when you see the pair 1,0 and a zero bit
> >>when you see the pair 0,1 (and nothing otherwise), then you've
> >>got a provably unbiased output stream -- the bias has been scrubbed
> >>from the input -- by the technique of throwing away everyting that
> >>*isn't* unbiased, broadly speaking.
> >
> >I believe that algorithm is attributed to Knuth.
>
> Before that, it was attributed to Von Neumann. it's an old one.

In electronics I believe this is called "edge detection" instead of "level detection". The transform from 01/10 to 0/1 takes in an edge and yeilds a level. Signal processing terminology also deals with these transforms, but their lexicon is irrelevant because we're dealing with independent events as opposed to impulse responses where dependence is presumed.

>
>
> >This all sounds good on the surface but I am not convinced at this
> >point. For example, there are sequences from a TRNG that are heavily
> >biased, like 111...1 and 000...0, yet the scheme above does not seem
> >to be able to produce those sequences. Can that anti-skewing technique
> >above generate all possible sequences including 111...1 and 000...0
> >with their expected probability based on their length? Is so, please
> >show us how.
>
> Certainly. Remember that the symbol (1) is generated from a subsequence
> of 10 (or 1110, 0010, etc... in fact, by any sequence of the form
> [(00)+(11)*10]. So any sequence of the form [(00)+(11)*10]*
> will "debiasify" into 11111...11...
>
> As to proof that the expected probability is based on the length;
> do it by case analysis and induction.
>
> There are four cases for the first four bits; as expressed above :
>
> >>The probability of getting the sequence 1, 0 is p * (1-p).
> >>The probability of getting the sequence 0, 1 is (1-p) * p.
>
>   The probability of getting the sequence 1, 1 is p*p.
>   The probability of getting the sequence 0, 0 is (1-p)*(1-p).
>
> Note : only the first two generate output.
>
> Assume for induction that all (output) sequences of length K or less
> are printed uniformly with probability proportional to their length.
> (The base case of length zero sequences is, literally, trivial).
> Assume for contradiction, wolog, that the sequence X1 is more
> probable than the sequence X0. This means that it's more likely
> that a 1 was printed than a 0. But *this*, in turn, means that
> the sequence [(00)+(11)*10] was *generated* more probably than
> the sequence [(00)+(11)*01]. The prefix probability is the same
> in both cases, so the only possible source of such difference is
> in the probability of generating 10 vs. 01. But by case analysis,
> these probabilities are equal, hence the desired contraction is
> obtained.
>
>   -kitten

Subject: Re: hardRandNumbGen
Date: Sat, 13 Feb 1999 15:50:54 GMT
From: rcktexas@ix.netcom.com (R. Knauer)
Message-ID: <36c59ebb.8612624@nntp.ix.netcom.com>
References: <79vfc8$d9q$1@quine.mathcs.duq.edu>
Newsgroups: sci.crypt
Lines: 42

On 11 Feb 1999 15:42:48 -0500, juola@mathcs.duq.edu (Patrick Juola)
wrote:

>>A TRNG must be capable of generating all possible finite sequences
>>equiprobably. If it is biased, then is it not doing that. Anti-skewing
>>procedures do not generate those sequences that under-representted
>>because of the bias.

>Um... This is simply incorrect. That's *exactly* what anti-skewing
>procedures do.
>
>Think of it this way. Assume you have a biased, but independent, bit
>source that generates 1s with probability p > 0.5. Consider two
>successive bits, x, y.
>
>The probability of getting the sequence 1, 0 is p * (1-p).
>The probability of getting the sequence 0, 1 is (1-p) * p, which is
>identical to the above.
>
>So if you output a 1 bit when you see the pair 1,0 and a zero bit
>when you see the pair 0,1 (and nothing otherwise), then you've
>got a provably unbiased output stream -- the bias has been scrubbed
>from the input -- by the technique of throwing away everyting that
>*isn't* unbiased, broadly speaking.

I have a further objection to this method.

If this method produced perfectly secure numbers, then it could be
applied to the output of any PRNG to produce perfectly secure random
numbers. But we know this is impossible in general.

I still think that correlation and bias are completely separate
concepts. You may be able to anti-skew a sequence, but that won't
remove correlation.

Bob Knauer

"The one thing every man fears is the unknown. When presented with this
scenario, individual rights will be willingly relinquished for the guarantee
of their well being granted to them by their world government."
--Henry Kissinger

Subject: Re: hardRandNumbGen
Date: 15 Feb 1999 09:19:27 -0500
From: juola@mathcs.duq.edu (Patrick Juola)
Message-ID: <7a9adf$dj9$1@quine.mathcs.duq.edu>
References: <36c59ebb.8612624@nntp.ix.netcom.com>
Newsgroups: sci.crypt
Lines: 38

In article <36c59ebb.8612624@nntp.ix.netcom.com>,
R. Knauer <rcktexas@ix.netcom.com> wrote:
>On 11 Feb 1999 15:42:48 -0500, juola@mathcs.duq.edu (Patrick Juola)
>wrote:
>
>>>A TRNG must be capable of generating all possible finite sequences
>>>equiprobably. If it is biased, then is it not doing that. Anti-skewing
>>>procedures do not generate those sequences that under-representted
>>>because of the bias.
>
>>Um... This is simply incorrect. That's *exactly* what anti-skewing
>>procedures do.
>>
>>Think of it this way. Assume you have a biased, but independent, bit
>>source that generates 1s with probability p > 0.5. Consider two
>>successive bits, x, y.
>>
>>The probability of getting the sequence 1, 0 is p * (1-p).
>>The probability of getting the sequence 0, 1 is (1-p) * p, which is
>>identical to the above.
>>
>>So if you output a 1 bit when you see the pair 1,0 and a zero bit
>>when you see the pair 0,1 (and nothing otherwise), then you've
>>got a provably unbiased output stream -- the bias has been scrubbed
>>from the input -- by the technique of throwing away everyting that
>>*isn't* unbiased, broadly speaking.
>
>I have a further objection to this method.
>
>If this method produced perfectly secure numbers, then it could be
>applied to the output of any PRNG to produce perfectly secure random
>numbers. But we know this is impossible in general.

No. This method produces perfectly unbiased numbers *IF* the
underlying bit sequence is independent. PRNGs, not being
independent, can't be improved by this.

-kitten

Subject: Re: hardRandNumbGen

Date: Mon, 15 Feb 1999 15:56:41 GMT
From: rcktexas@ix.netcom.com (R. Knauer)
Message-ID: <36c84226.10575306@nntp.ix.netcom.com>
References: <7a9adf$dj9$1@quine.mathcs.duq.edu>
Newsgroups: sci.crypt
Lines: 31

On 15 Feb 1999 09:19:27 -0500, juola@mathcs.duq.edu (Patrick Juola) wrote:

>No. This method produces perfectly unbiased numbers *IF* the
>underlying bit sequence is independent. PRNGs, not being
>independent, can't be improved by this.

Let me attempt to reiterate this and your earlier post.

If a RNG produces finite sequences independently, but they are not equidistributed, then the Knuth method of anti-skewing will cause the output to become equidistributed - which then brings the RNG into compliance with the specifications for a TRNG.

Is that correct?

If so, then this should be part of the design of a TRNG. One advantage is that if the output is shorted or pulled up, there will be no final output since there will be no dibits of the kind 01 or 10. That means that the TRNG will be self-diagnosing in that regard.

Bob Knauer

"Of all tyrannies, a tyranny exercised for the good of its victims may be the most oppressive. It may be better to live under robber barons than under omnipotent moral busybodies. The robber baron's cruelty may sometimes sleep, his cupidity may at some point be satiated; but those who torment us for our own good will torment us without end, for they do so with the approval of their consciences."
--C.S. Lewis

Subject: Re: hardRandNumbGen
Date: 15 Feb 1999 13:11:35 -0500
From: juola@mathcs.duq.edu (Patrick Juola)
Message-ID: <7a9o0n$e7u$1@quine.mathcs.duq.edu>
References: <36c84226.10575306@nntp.ix.netcom.com>
Newsgroups: sci.crypt
Lines: 44

In article <36c84226.10575306@nntp.ix.netcom.com>,
R. Knauer <rcktexas@ix.netcom.com> wrote:
>On 15 Feb 1999 09:19:27 -0500, juola@mathcs.duq.edu (Patrick Juola)
>wrote:
>
>>No. This method produces perfectly unbiased numbers *IF* the
>>underlying bit sequence is independent. PRNGs, not being
>>independent, can't be improved by this.
>
>Let me attempt to reiterate this and your earlier post.
>
>If a RNG produces finite sequences independently, but they are not
>equidistributed, then the Knuth method of anti-skewing will cause the
>output to become equidistributed - which then brings the RNG into
>compliance with the specifications for a TRNG.
>
>Is that correct?

Yes.

>If so, then this should be part of the design of a TRNG.

Not necessarily. This technique is sufficient but not necessary for removing bias, and there may be other, more appropriate techniques depending on your needs. For example, this technique throws away approximately 1/2 of the generated bits, which means that your random number generator needs to generate bits in excess of twice the needed volume. This could rightly be objected to as inefficient.

A further objection is that the number of bits that may need to be generated and thrown away are unbounded, and as such this would be inappropriate to use in a real-time system where response time is required to be faster than a certain threshhold. In plain English, I may be unwilling to wait several seconds before my RNG spits out any data.

Furthermore, if you are sufficiently confident that your generator is unbiased, such a technique is redundant.

So this is an engineering question, and not a formal requirement.

-kitten

Subject: Re: hardRandNumbGen
Date: Mon, 15 Feb 1999 20:03:47 GMT
From: rcktexas@ix.netcom.com (R. Knauer)
Message-ID: <36c87a91.25018835@nntp.ix.netcom.com>
References: <7a9o0n$e7u$1@quine.mathcs.duq.edu>
Newsgroups: sci.crypt
Lines: 48

On 15 Feb 1999 13:11:35 -0500, juola@mathcs.duq.edu (Patrick Juola)
wrote:

>Not necessarily. This technique is sufficient but not necessary
>for removing bias, and there may be other, more appropriate
>techniques depending on your needs. For example, this technique
>throws away approximately 1/2 of the generated bits, which means
>that your random number generator needs to generate bits in excess
>of twice the needed volume. This could rightly be objected to
>as inefficient.

>A further objection is that the number of bits that may need to
>be generated and thrown away are unbounded, and as such this would
>be inappropriate to use in a real-time system where response time
>is required to be faster than a certain threshhold. In plain English,
>I may be unwilling to wait several seconds before my RNG spits out
>any data.

>Furthermore, if you are sufficiently confident that your generator
>is unbiased, such a technique is redundant.

>So this is an engineering question, and not a formal requirement.

Until another proveably secure anti-skewing technique can be
identified, it has the advantage that it works. And since this
discussion is mostly theoretical regarding the proveable security of
the OTP system, the fact that such a generator would not be useable
for a real time stream cipher is not relevant.

Regarding non-TRNG streams (such as text streams), one thing that I
forgot to ask when we were discussing decorrelation techniques is
whether those schemes proposed, such as CRC hash or the LZ77
compression algorithm you suggested, also remove bias?

Do such decorrelation procedures also remove bias, or is it necessary
to run an anti-skewing technique on the stream? If so, should it be
run before or after the decorrelation procedure?

Bob Knauer

"Of all tyrannies, a tyranny exercised for the good of its victims may
be the most oppressive. It may be better to live under robber barons
than under omnipotent moral busybodies. The robber baron's cruelty may
sometimes sleep, his cupidity may at some point be satiated; but those
who torment us for our own good will torment us without end, for they
do so with the approval of their consciences."
--C.S. Lewis

Subject: Re: hardRandNumbGen
Date: Mon, 08 Feb 1999 19:28:59 GMT
From: ritter@io.com (Terry Ritter)
Message-ID: <36bf3adb.4721173@news.io.com>
References: <36bef38f.9237422@nntp.ix.netcom.com>
Newsgroups: sci.crypt
Lines: 82

On Mon, 08 Feb 1999 15:01:01 GMT, in
<36bef38f.9237422@nntp.ix.netcom.com>, in sci.crypt
rcktexas@ix.netcom.com (R. Knauer) wrote:

>[...]
>I realize that an actual TRNG will not be perfect, but I believe it
>can be made to specification such that it will not cause any
>significant amount of information to "leak".

I would say that to the extent that a source has some "true"
randomness, information leakage has no consequence.

If we can make all of our reported values dependent upon enough TRNG
values, we can assure (statistically!!) to a high probability that
each reported value is independent.

>That may also be the case for some stream ciphers, such as using the
>least significant bits of text and hashing them to high entropy
>density. Even though the latter is not a nondeterministic TRNG, it may
>not leak cause enough information to leak for practical purposes.

This may be combining two concerns: first to acquire the uniqueness
which exists in expression, and second to prevent the identification
of the presumably known source text. The second we can achieve by
throwing away information, and the first is best achieved by using the
larger character representation, and not just the lsb's.

>My concern is that statistical tests cannot be used to determine
>whether ANY method of random number generation is suitable for the OTP
>system. All it can be used for is to reject those RNGs that do not
>pass, and then only probabilistically.

I think reality is the other way around: I think we can reject bad generators to a very high probability, if we can detect them.

Often, statistical tests will set a PASS / FAIL bound on the statistic sequence from a generator and get a statistic which we can interpret as a "pass" or "fail." The probabilistic part of this is that the *ideal* random generator will "fail" about 1 percent of the time (or whatever level we select as failure).

In contrast, a "bad" generator (one with a measurable error) will fail very badly virtually all of the time. Of course, a bad generator which we cannot detect appears to be a good generator, and we will reject it 1 percent of the time, just like any other good generator.

>The only way you can certify a
>TRNG for use with the OTP system is to do a full audit of the
>generation method.

I agree.

>[...]
>I believe it is possible to build a TRNG that can be be certified to
>be properly working based on subsystem diagnostics and the knowledge
>of the random process that gives it its nondeterminancy - without
>having to resort to statistical tests on the output sequences.

Of course, the way we certify the TRNG is to run *statistical* tests on the randomness source. (There is not much use in trying to test the source on the basis of processed output sequences.) These are true statistical tests, and we only get probable results. But subsequent processing can accumulate randomness and throw away information, so even imperfect generators can be very usable.

We seek to certify, to high probability, using statistical tests, that a complex physical process is producing the results we expect. In this way we certify the detection machinery, and then depend upon the assumed characteristics of the physical source as the generator of randomness.

Terry Ritter ritter@io.com http://www.io.com/~ritter/
Crypto Glossary http://www.io.com/~ritter/GLOSSARY.HTM

Subject: Re: hardRandNumbGen
Date: Mon, 08 Feb 1999 20:54:11 GMT
From: rcktexas@ix.netcom.com (R. Knauer)
Message-ID: <36bf4dcb.32337138@nntp.ix.netcom.com>
References: <36bf3adb.4721173@news.io.com>
Newsgroups: sci.crypt
Lines: 24

On Mon, 08 Feb 1999 19:28:59 GMT, ritter@io.com (Terry Ritter) wrote:

>This may be combining two concerns: first to acquire the uniqueness
>which exists in expression, and second to prevent the identification
>of the presumably known source text. The second we can achieve by
>throwing away information, and the first is best achieved by using the
>larger character representation, and not just the lsb's.

I do not understand what you have just said wrt the LSB's. Why would using larger character representations be preferably to using LSBs?

>Of course, a bad generator
>which we cannot detect appears to be a good generator, and we will
>reject it 1 percent of the time, just like any other good generator.

That was exactly my point with regard to bad generators.

Bob Knauer

"The world is filled with violence. Because criminals carry guns,
we decent law-abiding citizens should also have guns. Otherwise
they will win and the decent people will loose."
--James Earl Jones

Subject: Re: hardRandNumbGen
Date: Tue, 09 Feb 1999 04:06:24 GMT
From: ritter@io.com (Terry Ritter)
Message-ID: <36bfb43d.2850225@news.io.com>
References: <36bf4dcb.32337138@nntp.ix.netcom.com>
Newsgroups: sci.crypt

Lines: 35

On Mon, 08 Feb 1999 20:54:11 GMT, in
<36bf4dcb.32337138@nntp.ix.netcom.com>, in sci.crypt
rcktexas@ix.netcom.com (R. Knauer) wrote:

>On Mon, 08 Feb 1999 19:28:59 GMT, ritter@io.com (Terry Ritter) wrote:
>
>>This may be combining two concerns: first to acquire the uniqueness
>>which exists in expression, and second to prevent the identification
>>of the presumably known source text. The second we can achieve by
>>throwing away information, and the first is best achieved by using the
>>larger character representation, and not just the lsb's.
>
>I do not understand what you have just said wrt the LSB's. Why would
>using larger character representations be preferably to using LSBs?

There is simply more information present in all the bits than in just
the LSB. This allows us to throw away more in the hashing.

By using only LSB's, we throw away the rest of each character. But
this is information which is not well mixed. So we may be throwing
away bits which are more variable -- and thus more valuable -- than
the lsb's.

Even when characters have an overall "entropy" of less than a bit,
there is no particular reason to expect to find the entropy
concentrated in the LSB's. The LSB represents part of the coding; it
has nothing to do with the entropy transported by the variation in
usage of that coding.

Terry Ritter ritter@io.com http://www.io.com/~ritter/
Crypto Glossary http://www.io.com/~ritter/GLOSSARY.HTM

Subject: Re: hardRandNumbGen
Date: Tue, 09 Feb 1999 13:20:57 GMT
From: rcktexas@ix.netcom.com (R. Knauer)
Message-ID: <36c03545.6590156@nntp.ix.netcom.com>
References: <36bfb43d.2850225@news.io.com>
Newsgroups: sci.crypt
Lines: 24

On Tue, 09 Feb 1999 04:06:24 GMT, ritter@io.com (Terry Ritter) wrote:

>Even when characters have an overall "entropy" of less than a bit,
>there is no particular reason to expect to find the entropy
>concentrated in the LSB's. The LSB represents part of the coding; it
>has nothing to do with the entropy transported by the variation in
>usage of that coding.

I suppose the problem I am having is that if you include all the bits
of the text characters, you will be introducing bias into the stream.
If you use only the first 128 ASCII characters, the most significant
bit will always be 0, so there will be a bias towards more 0s than 1s.

Do you assume that some kind of anti-skewing procedure has been
employed prior to hashing the data? Or is anti-skewing even necessary
in what you are proposing? Does hashing remove bias?

Bob Knauer

"The world is filled with violence. Because criminals carry guns,
we decent law-abiding citizens should also have guns. Otherwise
they will win and the decent people will loose."
--James Earl Jones

Subject: Re: hardRandNumbGen
Date: Tue, 09 Feb 1999 16:27:44 +0100
From: Mok-Kong Shen <mok-kong.shen@stud.uni-muenchen.de>
Message-ID: <36C053F0.38A1949B@stud.uni-muenchen.de>
References: <36bef38f.9237422@nntp.ix.netcom.com>
Newsgroups: sci.crypt
Lines: 12

R. Knauer wrote:
>

> A TRNG has a specific definition - it must be capable of generating
> all possible finite sequences equiprobably. The best example of a TRNG
> is a fair coin toss.

I believe lots of people would be very happy if you could tell them
how to obtain a fair coin! Isn't evident now that one can never get
an ideal OTP?

M. K. Shen

Subject: Re: hardRandNumbGen
Date: Tue, 09 Feb 1999 18:41:17 GMT
From: rcktexas@ix.netcom.com (R. Knauer)
Message-ID: <36c07fdb.2272868@nntp.ix.netcom.com>
References: <36C053F0.38A1949B@stud.uni-muenchen.de>
Newsgroups: sci.crypt
Lines: 31

On Tue, 09 Feb 1999 16:27:44 +0100, Mok-Kong Shen
<mok-kong.shen@stud.uni-muenchen.de> wrote:

>> A TRNG has a specific definition - it must be capable of generating
>> all possible finite sequences equiprobably. The best example of a TRNG
>> is a fair coin toss.

>I believe lots of people would be very happy if you could tell them
>how to obtain a fair coin! Isn't evident now that one can never get
>an ideal OTP?

I meant that coin toss system as an analogy.

I do not believe that chaotic classical systems can be proven to be
totally random. The reason is that they are computable. Only certain
Quantum Mechanical systems are proven to be totally random. Certain
Quantum Mechanical processes are uncomputable.

For example, the spontaneous emission that occurs in certain kinds of
radioactive decay is totally random in time - i.e., the time of any
particular decay is uncomputable. If anyone can demonstrate that it is
computable, they need to plan on the tux rental for their trip to
Stockholm.

Bob Knauer

"The world is filled with violence. Because criminals carry guns,
we decent law-abiding citizens should also have guns. Otherwise
they will win and the decent people will loose."
--James Earl Jones

Subject: Re: hardRandNumbGen
Date: Tue, 09 Feb 1999 12:21:04 -0700
From: "Tony T. Warnock" <u091889@cic-mail.lanl.gov>
Message-ID: <36C08AA0.3D4D5994@cic-mail.lanl.gov>
References: <36c07fdb.2272868@nntp.ix.netcom.com>
Newsgroups: sci.crypt
Lines: 79

R. Knauer wrote:

> On Tue, 09 Feb 1999 16:27:44 +0100, Mok-Kong Shen
> <mok-kong.shen@stud.uni-muenchen.de> wrote:
>
> >> A TRNG has a specific definition - it must be capable of generating
> >> all possible finite sequences equiprobably. The best example of a TRNG
> >> is a fair coin toss.
>
> >I believe lots of people would be very happy if you could tell them
> >how to obtain a fair coin! Isn't evident now that one can never get
> >an ideal OTP?
>
> I meant that coin toss system as an analogy.
>
> I do not believe that chaotic classical systems can be proven to be
> totally random. The reason is that they are computable. Only certain
> Quantum Mechanical systems are proven to be totally random. Certain
> Quantum Mechanical processes are uncomputable.
>
> For example, the spontaneous emission that occurs in certain kinds of
> radioactive decay is totally random in time - i.e., the time of any
> particular decay is uncomputable. If anyone can demonstrate that it is
> computable, they need to plan on the tux rental for their trip to
> Stockholm.
>
> Bob Knauer
>
> "The world is filled with violence. Because criminals carry guns,
> we decent law-abiding citizens should also have guns. Otherwise
> they will win and the decent people will loose."
> --James Earl Jones

Your example of coin tossing is not bad though. Some simulations (with a two
dimensional 4-sided die, the results should extrapolate to coins) show that
if the die is dropped from high enough, and that there is enough
elastisitcity in the floor, the probability of falling on any side is 1/4
within any epsilon of a starting orientation. If you bounce a die hard
enough, it acts random. The craps procotol in casinos reflect this. I coin,
flipped high, bounced at least once (craps require twice) ought to do quite
well. Of course this assumes a reasonably symmetrical coin. You ought to get
50% eagles and 50% buildings. von Neumann's bias removal can be used improve
results. Of course this is really slow.

Tony

news fodder:0110111001011101111000100110101011110011011110111110000100011001 0

1001110100101011011010111110001100111010110111110011101111101111110000010000 1

1000101000111001001001011001101001111010001010011010101010111011001011011011 10

1011111100001100011100101100111101001101011101101101111110001110011101011101 1

1111001111011111101111110000001000001100001010000111000100100010110001101000 111

1001000100100110010101001011100110010011011001110100111110100001010001101001 0

1010011101010010101011010110101011110110001011001101101010110111011100101110 1

1011110101111111000001100001110001011000111100100110010111001101100111110100 0

1101001110101011010111101100110110111011101101111110000111000111100101110011

1110100111010111101101110111111100011110011111010111101111111001111101111111 0

1111111100000000100000011000001010000011100001001000010110000110100001111000 1000

1000100110001010100010111000110010001101100011101000111110010000100100 01

1001001010010011100101001001010110010110100101111001100010011001100110 11010

1001101110011100100111011001111010011111101000001010000110100010101000 11

1010010010100101101001101010011110101000101010011010101010101011101011 00

1010110110101110101011111011000010110001101100101011001110110100101101 0110110110

1011011110111000101110011011110101011101110111100101111011011111010111 1111000000

1100000111000010110000111100010011000101110001101100011111001000110010 01

Subject: Re: hardRandNumbGen
Date: Mon, 08 Feb 1999 19:28:21 GMT
From: ritter@io.com (Terry Ritter)
Message-ID: <36bf3aca.4703677@news.io.com>
References: <79mre8$mni@b.stat.purdue.edu>
Newsgroups: sci.crypt
Lines: 28

On 8 Feb 1999 09:13:28 -0500, in <79mre8$mni@b.stat.purdue.edu>, in
sci.crypt hrubin@b.stat.purdue.edu (Herman Rubin) wrote:

>[...]
>For cryptography, the criterion is vulnerability of the
>transmitted message. For simulation, the criterion is the accuracy
>of the calculated results. In neither of these case is the source
>of the "random numbers" important, unless it causes the criterion
>not to be met.

I agree with this. In the end we get a sequence. The source of the
sequence does not matter to the cipher.

But as I see it, "the" important quality for a cryptographic sequence
is the independence of each value. Complete independence simply
cannot be tested. There are too many ways a sequence can have
dependence to test them all statistically.

And if we want each value that we report to be even *almost*
independent of all other reported values, we are not going to be using
any PRNG's.

Terry Ritter ritter@io.com http://www.io.com/~ritter/
Crypto Glossary http://www.io.com/~ritter/GLOSSARY.HTM

Subject: Re: hardRandNumbGen
Date: Fri, 29 Jan 1999 16:20:20 +0100
From: Mok-Kong Shen <mok-kong.shen@stud.uni-muenchen.de>
Message-ID: <36B1D1B4.3AE4762@stud.uni-muenchen.de>
References: <36b0f08a.10281173@nntp.ix.netcom.com>
Newsgroups: sci.crypt
Lines: 32

R. Knauer wrote:
>

> Your comments are directed at showing reasons for suspecting that a
> generator is not random. What do you do if your statistical tests
> indicate that it IS random?

Since you have been all the time disclaiming the usefulness of
statistical tests, the formulation of your question surprises me.
For it possibly indicates that you are rather unfamiliar with
such tests (apology if I am wrong). With statistical means you
can NEVER expect to 'prove' something 'is' or 'is not' (in the

absolute sense). Elsewhere Parick Juola has given good exposition of the essence of statistics. With my very humble knowledge I certainly can't do better. But perhaps I could attempt to give the following as a supplement to the topic of what a statistical test performs: One has a hypothesis H_0 and wants to investigate it statistically. One takes a sample. Given a confidence level alpha, one evaluate the value t of a test function T corresponding to the hypothesis and see if t is in the critical domain D of T for the given value alpha. If yes, H_0 is rejected (at the given confidence level). Otherwise one can only conclude that result of the test does not contradict H_0 (but it does NOT prove H_0!)

So you see people doing statistical tests are very prudent and very modest ones. They never say anything categorically ('for sure', IS or IS NOT) but rather say 'something is fairly unlikely to be true' or 'there is not enough evidence to believe that it is false'. (Compare such categorical statements as 'anything from software IS not crypto-grade' and 'everything from hardware IS crypto-grade'.)

M. K. Shen

Subject: Re: hardRandNumbGen
Date: Tue, 26 Jan 1999 08:18:01 -1000
From: handWave <real9@complex9.net>
Message-ID: <36AE06D9.384@complex9.net>
References: <36ADB62D.E681674F@stud.uni-muenchen.de>
Newsgroups: sci.crypt
Lines: 58

Mok-Kong Shen wrote:
>
> handWave wrote:
> >
>
> > > Even the presmption that the output would pass statistical tests is
> > > questionable. One famous gafffe in PRNG design was Knuth's composite
> > > generator, which he called superrandom. Unfortunately it was a closed loop
> > > design.
> >
> > It was a computer program.
>
> Having previously taken part in discussions in several threads of
> this group on random number generations, I doubt nevertheless that
> I have really known an answer to the following question:
>
> If I have two sources of randomness, one software and one hardware,
> both passing all statistical tests I apply equally well, why should
> I choose one source in preference to the other?

Consider than many situations RNGs are used for. In two dimensions there are four scenarios:

1 You want repeatable outputs or you want unpredictable outputs.
2 You use the RNG is a vaults with trusted guards or you use it in a college dormatory where you made some energetic enemies.

The preferences become clearer in those situations.

>And if additionally
> I don't know which sequence I get is from software and which is from
> hardware? (Compare the Turing test.) How does the origin of the
> sequence affect the workload of the analyst,

In that case you may chose either one with a chance of making a bad choice, depending on the future situational dimension in which it will be used. Random numbers sometimes seem nonrandom, hence lottery players sometimes find a lucky number or see a pattern.

>if the software
> generation process involves so many parameters that for combinatorical
> reasons he has no chance of directly dealing with them but has
> to try to look instead for possible regularities/irregularities in
> the sequence itself and, by assumption, the sequences from the
> two sources are of equal statistical quality? (Note that the
> hardware source is (in my humble opinion) unpredictable simply
> because there are so many participating 'parameters' that the
> 'summation' (the end product) becomes unpredictable, cf. the casting
> of a dice.)
>
> M. K. Shen

The scenario in which the RNG will be used makes the choice clearer. Pick a purpose: repeatability for stream ciphers, then use a PRNG. For key generation use a RNG based on hardware. If you are in the college dorm with enemies who may interfere remotely, then a thermal noise generator from an avalanche diode with a high gain amplifier breadboared using wire-wrapped, unshielded circuits is a bad implementation.

Subject: Re: hardRandNumbGen
Date: Tue, 26 Jan 1999 18:24:51 +0100
From: Mok-Kong Shen <mok-kong.shen@stud.uni-muenchen.de>
Message-ID: <36ADFA63.61CC6CAB@stud.uni-muenchen.de>

References: <36AE06D9.384@complex9.net>
Newsgroups: sci.crypt
Lines: 21

handWave wrote:
>

> The scenario in which the RNG will be used makes the choice clearer. Pick
> a purpose: repeatability for stream ciphers, then use a PRNG. For key
> generation use a RNG based on hardware. If you are in the college dorm
> with enemies who may interfere remotely, then a thermal noise generator
> from an avalanche diode with a high gain amplifier breadboared using
> wire-wrapped, unshielded circuits is a bad implementation.

I think the following scenario may be interesting: I am the user
of an application with certain security expectations. Some one
offers me two sequences generated from two different sources
(I am not told which is which). I want to decide which one to use.
I have all statistical test tools available. How am I going to make
the decision? (This is independent of how I am going to use the
sequence, I suppose. I simply want the one with better 'crypto
strength', i.e. higher un-predictability. Maybe I'll use the
sequence as kind of one-time pad (never mind the terminology here).)

M. K. Shen

Subject: Re: hardRandNumbGen
Date: Tue, 26 Jan 1999 19:36:01 GMT
From: ritter@io.com (Terry Ritter)
Message-ID: <36ae1916.10719434@news.io.com>
References: <36ADB62D.E681674F@stud.uni-muenchen.de>
Newsgroups: sci.crypt
Lines: 108

On Tue, 26 Jan 1999 13:33:49 +0100, in
<36ADB62D.E681674F@stud.uni-muenchen.de>, in sci.crypt Mok-Kong Shen
<mok-kong.shen@stud.uni-muenchen.de> wrote:

>[...]
>If I have two sources of randomness, one software and one hardware,
>both passing all statistical tests I apply equally well, why should
>I choose one source in preference to the other?

This of course depends upon the desired use; here we assume that the
use will be as some sort of cryptographic value. That means the
sequence must be "unpredictable." And we assume in cryptography that
the design itself is available for analysis, with a large amount of
the resulting sequence.

Can a software RNG remain "unpredictable" under these conditions?
Presumably the answer is "Yes," but we have long experience with the
confusion generators in stream cipher cryptography, where many, many,
apparently-strong RNG's have been broken. (Note that digital
state-machine constructions can be realized in either hardware or
software.)

In cryptography we can neither measure nor prove strength. We *can*
prove or demonstrate weakness, and we call such a proof a "break."
But finding a break is often a difficult exercise, so we cannot simply
assume that the absence of a break means that none are to be found.

So how are we to trust a software state-machine RNG to produce our
unpredictable values? One way is to use a well-known cryptographic
hash or cipher to protect the original values. Another way is to
measure molecular-level events which are "known" to be unpredictable.

The problem with molecular-level events is that they are very, very
tiny, and that many other signals which we normally ignore are of a
similar magnitude or even larger. But if we *could* measure such
events, we would have a good theoretical basis for asserting
strength (in the sense of "unpredictability"), which is otherwise
generally unavailable in cryptography.

Now, no real device is going to be ideal, and presumably there will
always be some possibility of hidden "weakness." So as a very basic
first step, we must be very sure that we actually are measuring and
reporting molecular-level randomness, rather than some other signal.
One way to do this is to arrange to "turn off" the randomness source,
and then inspect the output of the machine and see that it is quiet.
Then we can know our machine is indeed measuring the events that we
have just turned off.

Ideally, a randomness machine will allow us to conduct statistical
experiments on the randomness source itself (as opposed to a processed
"random" output). We have a strong theoretical understanding of
molecular-level randomness sources, so we know what the distributions
should look like under various conditions. Ideally, the machine will
allow us to change the conditions and collect statistics and see how
the compare to theory. Presumably, if the machine is working well,
there may be some deviation, but will largely produce the theoretical
results. If so, we can be fairly sure that we are actually measuring

the signal we hope to be measuring.

So, ideally, by tests we can confirm that we can detect noise, and that the noise has the structure we expect from theory. If so, we have a fairly strong argument that we are measuring "unknowable" randomness, and then only need to process it (typically in a hash, and CRC would be ideal) to obtain unknowable uniformly-distributed values.

>And if additionally
>I don't know which sequence I get is from software and which is from
>hardware? (Compare the Turing test.) How does the origin of the
>sequence affect the workload of the analyst,

Note that the goal of the analysis is to "break" (be able to predict) the generator. That obviously depends upon the generator.

>if the software
>generation process involves so many parameters that for combinatorical
>reasons he has no chance of directly dealing with them but has
>to try to look instead for possible regularities/irregularities in
>the sequence itself and, by assumption, the sequences from the
>two sources are of equal statistical quality?

The way generators are broken is through a detailed analysis of the generation process, which may involve statistical support. But, on their own, statistical tests are simply uninformed about the structure of the generator, and so cannot test correlations to the construction. But it is precisely the construction which we seek to break.

>(Note that the
>hardware source is (in my humble opinion) unpredictable simply
>because there are so many participating 'parameters' that the
>'summation' (the end product) becomes unpredictable, cf. the casting
>of a dice.)

That would be the complexity construction, which is closely related to the way we construct conventional ciphers. The problem with this is that most new ciphers turn out to be weak. So how can we trust this same construction when applied to RNG's?

Terry Ritter ritter@io.com http://www.io.com/~ritter/
Crypto Glossary http://www.io.com/~ritter/GLOSSARY.HTM

Subject: Re: hardRandNumbGen
Date: Wed, 27 Jan 1999 16:06:40 +0100
From: Mok-Kong Shen <mok-kong.shen@stud.uni-muenchen.de>
Message-ID: <36AF2B80.C844BA93@stud.uni-muenchen.de>
References: <36ae1916.10719434@news.io.com>
Newsgroups: sci.crypt
Lines: 40

Terry Ritter wrote:
>

> So how are we to trust a software state-machine RNG to produce our
> unpredictable values? One way is to use a well-known cryptographic
> hash or cipher to protect the original values. Another way is to
> measure molecular-level events which are "known" to be unpredictable.

I like however to associate that "known" with some degree of subjectivity (unless that "known" can be shown to be the same as "proved"). So I would say that the decision I referred to is not entirely free from subjectivity.

> >And if additionally
> >I don't know which sequence I get is from software and which is from
> >hardware? (Compare the Turing test.) How does the origin of the
> >sequence affect the workload of the analyst,
>
> Note that the goal of the analysis is to "break" (be able to predict)
> the generator. That obviously depends upon the generator.

In another follow-up I described more precisely the difficult situation of a user who has to choose among two given sequences without additional information as to the sources. I don't think that a choice based on objective criteria alone is (always) possible.

>
> The way generators are broken is through a detailed analysis of the
> generation process, which may involve statistical support. But, on
> their own, statistical tests are simply uninformed about the structure
> of the generator, and so cannot test correlations to the construction.
> But it is precisely the construction which we seek to break.

But the generator may under circumstances not be exactly known, namely if its design (its structure etc.) depends on a number of

parameters not known to the analyst. Brute-forcing these may be
imfeasible if there is a combinatorial explosion.

M. K. Shen

Subject: Re: hardRandNumbGen
Date: Tue, 26 Jan 1999 12:15:34 -0500
From: "Trevor Jackson, III" <fullmoon@aspi.net>
Message-ID: <36ADF835.5630B028@aspi.net>
References: <36ADC1FB.4212@complex9.net>
Newsgroups: sci.crypt
Lines: 73

handWave wrote:

> Trevor Jackson, III wrote:
>
> handWave wrote:
> > > I am glad you raised the "handwaves" metaphore, because handwaves are
> > > what toss coins. A complex person tosses a coin and you might think it
> > > is random. The oscillator RNG in this discussion is directly analogous
> > > to a coin toss in many ways. If a coin is not rotating (oscillating)
> > > it will fall back into the hand in the same position that it stated from.
> > > It is the rotation that helps the randomness, not only the complexity
> > > of the nervous system, the elasticity of the skin, and the trembling of
> > > the muscles. The rotation should be fast for best results. A juggler
> > > could become skilled at non-random coin tosses for one coin that
> > > rotates slowly. But if she tosses three coins with rapid rotation than
> > > it is likely that random results will occur. If a periodic wind is
> > > present in a coin toss, yes, it will influence the outcome, but the
> > > result will often be recognizable as a useful random throw, or a throw
> > > that was blown away. The same with this RNG.
> >
> > Human gestures are not a good foundation for system design. There are large,
> > industrial concerns that rely upon human-gesture-generated unpredictability.
> > Their interest is *not* statistical randomness as we find in simulations,
> > games, and Monte Carlo tests (in spite of the latter name). Their interest
> > is the same as ours: unpredicability. They are called casinos.
>
> The product I designed was evaluated for casinos by Bally, a potential
> customer.
>
> >
> > In spite of the fantastic efforts taken to eliminate predictability in games
> > of chance human gestures can still dominate the outcomes completely. I'm not
> > referring to shuffling cards systematically, but to rolling a roulette ball
> > against the wheel so precisely that out of 20 tries a human can obtain a
> > predicted outcome (slot 17) 10 times. 50% success. I think that constitutes
> > predictability.
>
> Yes, this is like the skilled juggler I described above. The analogy to a
> hardRandNumbGen is a skilled hacker who controls the power supply noise,
> the clock glitches, the radio beams so that the RNG becomes under his
> control. The chip designer must anticipate such antics, and prepare the
> module for lunar insertion.
>
> > The hand-eye coordination involved is of an extreme level, requiring decades
> > of practice to achieve. But it is real. The complexity of controlling a
> > roulette wheel appears to me to be far larger than that of a coin toss. Even
> > a fast one.
>
> I dispute this. A coin has one bit of output, a wheel has many bits in
> one toss. A wheel is a big target with a smaller bandwidth for RPMs. A
> coin has a wider bandwidth, perhaps 1hz to 50 hz, a wheel, from .1 hz to
> .5 hz on the initial spin. A coin may be tossed from a rooftop. Wheels
> would fracture under such conditions.

Then we disagree strongly. My statement was not based on the degrees of freedom of
the two systems, nor on unanticipated operations on the hardware such as
defenestration. I was referring only to the degree of practice required by a human
to reach a particular skill level. For instance, to get a 10% advantage given the
standard payoff odds. I doubt an average person could control a roulette wheel
that well after a year of practice. OTOH, I think an average person could reach
that level of skill with a silver dollar after a few days or a week.

Clearly the coin and the wheel are part of the physical universe and thus subject
to the same set of arcane and esoteric influences. But the coin is much easier to
control than the wheel. So we have to assume that a coin is not as good a source
of entropy because it is too easy for an external influence to inject bias into the
results.

[mega snip]

As for the rest of your response, I can only harumph and murphle, which are not
productive of enlightenment.

Subject: Re: hardRandNumbGen
Date: Mon, 25 Jan 1999 12:29:42 -0600
From: Medical Electronics Lab <rosing@physiology.wisc.edu>
Message-ID: <36ACB816.35EF@physiology.wisc.edu>
References: <36AC8363.6D55@complex.net>

Newsgroups: sci.crypt
Lines: 21

.´¯`...´¯`...´¯`...´¯`...´¯`...´¯`.. wrote:
[...]
> Radioactive decay is also a large signal RNG. It may be considered
> to be both digital and analog, as this RNG may be.

It depends on how you measure it. Using a smoke detector it's
a mighty damn weak signal. It takes about 1e5 amplification to
see the radiation noise and that involves lots of shielding from
outside sources as well as preventing feedback.

It's not likely that anyone would stick a radiation source onto
a processor either, so it has to be external. For the truely
paranoid this is a good thing, they can look at their hardware
and slap 'scopes on just to be sure everything is copasetic.

> Thank you for this polite discussion.

It has been fun reading, thank you too!

Patience, persistence, truth,
Dr. mike

Subject: Re: hardRandNumbGen
Date: Tue, 26 Jan 1999 02:56:36 -1000
From: handWave <real9@complex9.net>
Message-ID: <36ADBB84.775@complex9.net>
References: <36ACB816.35EF@physiology.wisc.edu>
Newsgroups: sci.crypt
Lines: 43

Medical Electronics Lab wrote:
>
> .´¯`...´¯`...´¯`...´¯`...´¯`...´¯`.. wrote:
> [...]
> > Radioactive decay is also a large signal RNG. It may be considered
> > to be both digital and analog, as this RNG may be.
>
> It depends on how you measure it. Using a smoke detector it's
> a mighty damn weak signal. It takes about 1e5 amplification to
> see the radiation noise and that involves lots of shielding from
> outside sources as well as preventing feedback.
>
> It's not likely that anyone would stick a radiation source onto
> a processor either, so it has to be external. For the truely
> paranoid this is a good thing, they can look at their hardware
> and slap 'scopes on just to be sure everything is copasetic.

Radioactive sources occur in dynamic RAMs, which cause soft errors: LARGE
SIGNALS from thorium and uranium traces in common aluminum. Tim May
showed this around 1980. The problem is not putting radioactive materials
into chip, the problem is keeping them out. Special processing has
reduced this problem in recent decades. If you have a 1979 DRAM laying
around, it would be a slow source of random numbers as a Large Signal
Random Number Generator. This has been discussed before. Just write all
ones to the DRAM, observe the soft errors that cause zeros, and use the
address and time of the bit flipped to seed your hash.

Or take your smoke detector and place the vitals in contact with the
surface of a modern DRAM: Large Random Signals would result in the big
memory array. Take two and call me in the morning.

.´¯`...´¯`...´¯`...´¯`...´¯`...´¯`.. handWave

> > Thank you for this polite discussion.
>
> It has been fun reading, thank you too!
>
> Patience, persistence, truth,
> Dr. mike

You're welcome, God Bless You!

Subject: Re: hardRandNumbGen
Date: Tue, 26 Jan 1999 09:33:00 -0000
From: "burt" <Lee.Grime@rdl.co.uk>
Message-ID: <36ad8bb3.0@nnrp1.news.uk.psi.net>
References: <36AC8363.6D55@complex.net>
Newsgroups: sci.crypt
Lines: 4

Have you done any Frequency domain analysis on the signals produced?? any
periodicity in the frequency domain?

Subject: Re: hardRandNumbGen
Date: Tue, 26 Jan 1999 02:46:01 -1000
From: handWave <real9@complex9.net>
Message-ID: <36ADB909.4F73@complex9.net>
References: <36ad8bb3.0@nnrp1.news.uk.psi.net>
Newsgroups: sci.crypt
Lines: 18

burt wrote:
>
> Have you done any Frequency domain analysis on the signals produced?? any
> periodicity in the frequency domain?

Our team tested the hardRandNumbGen using a spectrum analyser during 1984. I do not have any photos of spectra available now, 15 years later, because I resigned from that company many moons ago. We were looking for radiated signals that could be picked up from outside the chip, and we found a strong signal in the 500khz range, corresponding to the oscillator used to sample the faster waveforms. There were other, weaker frequencies detected, but I have no interesting facts to tell you about the spectrum.

As I recall, the two faster oscillators ran at about 5Mhz to 50Mhz as their frequencies were randomly modulated.

handWave

Subject: Re: hardRandNumbGen
Date: Tue, 26 Jan 1999 19:35:40 GMT
From: ritter@io.com (Terry Ritter)
Message-ID: <36ae1907.10704108@news.io.com>
References: <36AC8363.6D55@complex.net>
Newsgroups: sci.crypt
Lines: 283

On Mon, 25 Jan 1999 04:44:51 -1000, in <36AC8363.6D55@complex.net>, in
sci.crypt ".·´¯`···´¯`···´¯`···´¯`···´¯`···´¯`·."
<real@complex.net> wrote:

>On Sun, 24 Jan 1999 03:53:40 -1000, in <36AB25E4.2E7E@complex.net>, in
>sci.crypt <real@complex.net> sinewave wrote:
>
>>Terry Ritter wrote:
>[...]
>>That means that this "large signal" design is probably sensitive to
>>even tiny power and ground transients. It is going to be very hard to
>>distinguish the effects of "real" thermal noise from transient
>>feedback due to the structure of the circuit. So how can we have
>>confidence in the result? Statistical testing cannot distinguish
>>between "physical" and "pseudo" randomness.
>
>In the real world, it is not always possible to tell.

I have never heard IC designers claim that it is "not possible to tell" whether their design works. In fact, if we can't tell whether a design works or not, we generally consider that a design *failure*: If we can't test it, we can't trust it.

>Integrating
>a random number generator (RNG) on a commodity IC is similar to
>a manned expedition to MARS: they must take everything with them
>into that harsh environment that they will need. If the craft is
>buffeted by periodic winds, they do not have the luxury of calling
>back to base and saying, "Houston, you told us this was a vacuum,
>please make it a perfect vacuum, over". The RNG will encounter
>non-ideal electrical environments.

Every electronic circuit has requirements; we specify those requirements and then demand that the larger system achieve them if it expects the circuit to work. Conceivably, a low-noise section might have its own filtered analog power with a single-point ground, and separate digital power.

>It should have redundant systems

Yet we don't have, say, redundant adders on a processor. Why? One reason is that the result can be even *less* reliable. What we do is *test* the part, and if it doesn't work, we don't ship it. Which means we have to be able to test it.

>which are combined to give the final random number the best shot at
>being unpredictable, not perfect, but unpredictable.

If one is going to claim that a design achieves its goals due to thermal noise, one must at the very least be able to show a performance change between noise and no-noise. Combining the thermal noise detection with RNG feedback complexity generally prevents this, which means we can't test the very thing we claim to do.

>[...]

>The major source of randomness of this RNG is the unsynchronized
>nature of multiple oscillators with randomly changing frequencies.

First, it is in the nature of this sort of oscillator to synchronize to tiny transients as may occur from nearby state changes. So just how do we *know* that our oscillators *are* "unsynchronized"?

Next, the idea that the oscillators have "randomly changing frequencies" is what we need to prove or demonstrate, and not just assume. Even chaotic oscillators can follow predictable patterns.

>This
>is a large signal phenomenon, which cannot be accurately described
>mathematically.

Large signal phenomena are precisely those which are best described mathematically. It is the tiny signals (which must compete with thermal noise and transients from capacitive, inductive, and electromagnetic coupling) which are difficult to model well.

>Similar to a coin toss, many analog variables are
>involved. These continuous variations of many influences cause seeming
>randomness. If you can mathematically describe a human coin toss, then
>so you can with this RNG. But you cannot, and I cannot. That does not
>invalidate the usefulness of these seed generators, not in this
>century.

So I guess this is not a machine which provably detects random molecular events and conveys them to us in the larger world. It is instead something which looks quite a lot like yet another complex software PRNG.

>[...]
>>Easy calculations using the published results show that the effective
>>population of values is 1/4 the claimed ideal, which shows that the
>>design was not as good as you thought.
>
>Correct, that first version in that report had an XOR gate placed in
>a bad position, causing twice as many ones as zeros. The CIA alerted
>us to my mistake with that one gate. When removed, the results are
>much better. I still regret my mistake in that one gate placement.

Thank you for the confirmation. Usually I hear about my mistakes.

Apparently I was able to detect a problem in the design which your team and famous consultant did not, but which the CIA apparently also detected.

In other words, my analysis was contrary to your beliefs and those of your crypto expert, but was also correct.

So why are you not listening to me now?

>The ealier description was an illustration for some readers to examine.
>It was not an exhaustive explanation of the theory behind the design.
>I have now expanded upon the description, explaining the large
>signals as being analogous to coin tosses which must rotate due to
>a complex had waving motion. The complexity of my circuit design
>mimics, on a small scale, the complexities of the human hand wave
>and coin toss. The frequency changes in the design are the analogy
>of the hand motion. Thermal irregularities power supply variations
>also contribute to this hand motion.

That is a very unsatisfying explanation for an on-chip device composed of extremely-well-understood physical components and techniques.

>[...]
>I do not claim nobody can break this. I am presenting concepts to a
>wide reading audience. Some of these concepts are less sound than
>others, so the readers have the opportunity to judge various attepts
>to produce randomness in a harsh environment. I hope that they will
>fare better than I did.

And that is the same reason I confront those claims. Someday I may have to trust one of those systems.

>>The reasoning about this design is contradictory: Supposedly the
>>large signal design is "random" because it senses low-level noise.
>>Yet the circuit is supposedly suitable for a noisy digital chip
>>because it is a "large-signal" design. There is a fundamental problem
>>in making both claims at the same time.
>
>I have addressed this above. A large signal, digital oscillator has
>small noise on top of that.

And, to the extent that it is sensitive to "small noise," that part of its operation is no longer "large signal."

>The randomness is primarily based on the

>coin toss analogy.

Every binary sequence has 1's and 0's, but that does not make the generator physically random, or even pseudorandom.

>The thermal noise calculation first given is a
>secondary source of randomness.

And the first source would be?

>The periodic power supply noise
>will affect this design more in some ways than it would affect an
>analog circuit with well designed differential and common mode
>considerations.

I would say so, yes.

>But the ways periodic noise affects these circuits
>do not ruin the unpredictability of the resulting numbers.

If these stages are switching not based on thermal noise, but instead from nearby power transients from digital switching, they could appear to work, and yet be absolutely correlated to some apparently unrelated but physically-close device. And that could be predictable, if we are willing to invest the effort in finding the relationship. And once we do, of course, we can apply that knowledge to all examples of the device.

>I leave
>that discussion for another day.

Well :)

>[...]
>>But that approach is digital complexity, and not thermal randomness.
>>It can be simulated in software. It is PSEUDO-random. Maybe it is
>>strong, maybe not, but there is certainly no proof.
>
>It is analog complexity. I will give no proof today. Give me proof
>of coin tossing that does not involve complexity or strength..

This is the same sort of argument we very often hear from crypto newbies.

Unfortunately, it is not possible to prove -- or even measure -- cryptographic strength. It *is* possible to prove weakness by finding "breaks," but typically at great cost. That leaves the rather unsatisfying task of arguing strength, which is not working here.

>[...]
>>The obvious experiment, then, is to take the device to cryogenic
>>temperatures and see how it performs. If the output still has good
>>statistics, we can suspect that the output does not represent thermal
>>noise at all, but is just a complex digital system. Was such an
>>experiment performed?
>
>No. The circuit depends on many complex factors for randomness, as a
>coin toss does. In some imagined laboratory experiment, it is feasible
>to control all factors, causing non-random results. In commodity
>applications, Large Signal Random Number Generators are sometimes
>superior to small signal based generators and both may appear on a
>single IC.

But what would make you think "Large Signal Random Number Generators" are superior? Even the name is a deception which hides the fact that switching depends upon tiny noise-level values. The more you discuss this, the more it sounds like the sort of snake oil we see around her all the time.

>[...]
>>Even PSEUDO-random RNG's pass statistical tests. Those tests have
>>nothing to do with cryptographic unpredictability or "strength." Yet
>>strength is what you claim.
>
>Yes it is a strong source, as upcoming product releases are expected to
>show.

A press release shows us cryptographic strength?

If these circuits cannot demonstrate that they are detecting the thermal randomness they are said to use, there is no set of tests in the world which would be sufficient to show "strength."

>Just because old PRNGs pass some tests does not mean that new
>designs are bad, as you imply.

I imply no such thing. Perhaps you missed the point.

>>I think you have missed the distinction between unpredictable
>>randomness for cryptography, and ordinary statistical randomness.
>
>A PSRG may be depended upon to produce the same string under certain
>easy to arrange conditions. This RNG does the opposite of that.

That is what you need to prove, or to argue in a convincing way.

I argue that an RNG as you describe not only has digital state, it also has analog thermal state, direct electrical coupling from other power-using circuits (including each stage), and indirect coupling from nearby circuits. Once we account for all this state and interaction, I expect that we can largely predict the result.

The worst case would be that some stages trigger from some unexpected power transient, or even trigger off each other by unexpected coupling. In this case, prediction becomes much easier. Most dangerously, such a device might well seem "random" to all tests not specifically aware of the unexpected transient or coupling.

>Two
>sequential random numbers from this circuit would prove that to
>anyone who tests it, most of the time.

Testing the output cannot be sufficient to test the design. The engineering in circuit design is not about handwaves, it is about the attempt to absolutely understand and control device operation. So if we can't test it, we can't trust it.

Terry Ritter ritter@io.com http://www.io.com/~ritter/
Crypto Glossary http://www.io.com/~ritter/GLOSSARY.HTM

Subject: Re: hardRandNumbGen
Date: Tue, 26 Jan 1999 19:02:55 -0500
From: "Kazak, Boris" <bkazak@erols.com>
Message-ID: <36AE57AF.4C5C@erols.com>
References: <36ae1907.10704108@news.io.com>
Newsgroups: sci.crypt
Lines: 37

Terry Ritter wrote:
>
>
> Large signal phenomena are precisely those which are best described
> mathematically. It is the tiny signals (which must compete with
> thermal noise and transients from capacitive, inductive, and
> electromagnetic coupling) which are difficult to model well.
>

Let's be practical...

Consider such a simple system:

```
                HHHHHHHHHHHHHHHH
              HH               H   MMM
            HH                 H   MMMMM
            HH     OOOOOOOOOO  H   MMMMM
              HH  OOOOOOOOOOOO H   MMM
                HHHHHHHHHHHHHHHH
```

where HH is a Housing (just a glass or plastic bottle), OO are Objects (a pseudo-scientific baptism for 100-200 peas or beans), MM is a Microphone.
Now if we start rotating the Housing around its horizontal axis, the Objects will produce a loud Random Rattle, and the Microphone will transmit this rattle to the sound card. My questions are:

How many Objects are needed and what must be the speed of rotation that will assure the True Randomness?
What estimates can be given for Degree of Correlation and for Period of Repetition, depending on the system parameters?

The System is not patented, it is hereby placed in the public domain.

Respectfully BNK

Subject: Re: hardRandNumbGen
Date: Wed, 27 Jan 1999 07:27:34 -1000
From: handWave <shaken@stirred.com>
Message-ID: <36AF4C86.18CC@stirred.com>
References: <36AE57AF.4C5C@erols.com>
Newsgroups: sci.crypt
Lines: 56

Kazak, Boris wrote:
>
> Terry Ritter wrote:

```
> >
>
> >
> > Large signal phenomena are precisely those which are best described
> > mathematically.  It is the tiny signals (which must compete with
> > thermal noise and transients from capacitive, inductive, and
> > electromagnetic coupling) which are difficult to model well.
> >
> -------------------
> Let's be practical...
>
>    Consider such a simple system:
>
>                      HHHHHHHHHHHHHHHH
>                    HH               H   MMM
>                  HH                 H   MMMMM
>                  HH     OOOOOOOOOO  H   MMMMM
>                    HH  OOOOOOOOOOOO H   MMM
>                      HHHHHHHHHHHHHHHH
>
> where HH is a Housing (just a glass or plastic bottle), OO are Objects
> (a pseudo-scientific baptism for 100-200 peas or beans), MM is a
> Microphone.
>    Now if we start rotating the Housing around its horizontal axis,
> the Objects will produce a loud Random Rattle, and the Microphone will
> transmit this rattle to the sound card. My questions are:
>
>         How many Objects are needed

One.

>and what must be the speed of
> rotation that will assure the True Randomness?

If the bottle is smooth inside, rotating is not as good as shaking it.
Shaking by hand is like a coin toss: it is the result of a complex
process that has never been repeated exactly during human history.

>         What estimates can be given for Degree of Correlation and
> for Period of Repetition, depending on the system parameters?

Your sound card and software will need to comprehend the natural resonant
frequencies of the bottle and filter out that repeating soundwave. A
threshold for the clicks of a fall should be set to reject small resonant
sounds. Multiple bounces after a fall may have similarities to previous
bounces, so that should be rejected.

>
>     The System is not patented, it is hereby placed in the public
> domain.
>
>       Respectfully                        BNK
```

```
Subject: Re: hardRandNumbGen
Date: Wed, 27 Jan 1999 16:47:58 GMT
From: rcktexas@ix.netcom.com (R. Knauer)
Message-ID: &LT;36af42cb.24504355@nntp.ix.netcom.com>
References: &LT;36AF4C86.18CC@stirred.com>
Newsgroups: sci.crypt
Lines: 20

On Wed, 27 Jan 1999 07:27:34 -1000, handWave &LT;shaken@stirred.com>
wrote:

>Your sound card and software will need to comprehend the natural resonant
>frequencies of the bottle and filter out that repeating soundwave. A
>threshold for the clicks of a fall should be set to reject small resonant
>sounds. Multiple bounces after a fall may have similarities to previous
>bounces, so that should be rejected.

I suspect that all classical processes, even chaotic ones, suffer from
some kind of flaw as you describe. That's why I would only use quantum
processes for a TRNG.

Bob Knauer

"No Freeman shall ever be debarred the use of arms. The strongest
reason for the people to retain the right to keep and bear arms is,
as a last resort, to protect themselves against tyranny in government."
--Thomas Jefferson
```

```
Subject: Re: hardRandNumbGen
Date: Wed, 27 Jan 1999 02:35:56 -1000
From: handWave &LT;shaken@stirred.com>
Message-ID: &LT;36AF082C.3AA6@stirred.com>
References: &LT;36ae1907.10704108@news.io.com>
Newsgroups: sci.crypt
Lines: 39

Terry Ritter wrote:
```

>
> On Mon, 25 Jan 1999 04:44:51 -1000, in <36AC8363.6D55@complex.net>, in
> sci.crypt "..´¯`...´¯`...´¯`...´¯`...´¯`...´¯`.."
> <real@complex.net> wrote:
>
> >On Sun, 24 Jan 1999 03:53:40 -1000, in <36AB25E4.2E7E@complex.net>, in
> >sci.crypt <real@complex.net> sinewave wrote:
> >
> >>Terry Ritter wrote:

> >>Easy calculations using the publushed results show that the effective
> >>population of values is 1/4 the claimed ideal, which shows that the
> >>design was not as good as you thought.
> >
> >Correct, that first version in that report had an XOR gate placed in
> >a bad position, causing twice as many ones as zeros. The CIA alerted
> >us to my mistake with that one gate. When removed, the results are
> >much better. I still regret my mistake in that one gate placement.
>
> Thank you for the confirmation. Usually I hear about my mistakes.
>
> Apparently I was able to detect a problem in the design which your
> team and famous consultant did not, but which the CIA apparently also
> detected.
>
> In other words, my analysis was contrary to your beliefs and those of
> your crypto expert, but was also correct.
>
> So why are you not listening to me now?

I am listening. My silence for 24 hours does not mean I am not listening. And the consultant we used did not work on the RNG, he worked on the hash function, only. Your messages are beginning to repeat themselves, so I am not responding to these repeated comments. Thank you for a rational discussion of this subject. If you post more on this subject , I will be reading them, but not always responding.

..´¯`...´¯`...´¯`...´¯`...´¯`...´¯`..handWave

Subject: Re: hardRandNumbGen
Date: Sun, 31 Jan 1999 15:23:49 GMT
From: bo.doemstedt@mbox200.swipnet.se (Bo Dömstedt)
Message-ID: <36b47461.856793353@nntpserver.swip.net>
References: <36AC8363.6D55@complex.net>
Newsgroups: sci.crypt
Lines: 15

"someone" wrote:
>Integrating a random number generator (RNG) on a commodity
>IC is similar to a manned expedition to MARS: they must take
>everything with them into that harsh environment that they will need. If the
>craft is buffeted by periodic winds, they do not have the luxury of calling
>back to base and saying...
...etc...
Dear "someone", I would strongly recommend that
you go to Mars at earliest possible time. As for random number
generation, we sell them for $170...
http://www.protego.se/sg100_en.htm

Bo Dömstedt
Protego Information AB

Subject: Re: hardRandNumbGen
Date: Sun, 31 Jan 1999 12:04:45 -0500
From: "Kazak, Boris" <bkazak@erols.com>
Message-ID: <36B48D2D.6B25@erols.com>
References: <36b47461.856793353@nntpserver.swip.net>
Newsgroups: sci.crypt
Lines: 24

Bo Dömstedt wrote:
>
> "someone" wrote:
> >Integrating a random number generator (RNG) on a commodity
> >IC is similar to a manned expedition to MARS: they must take
> >everything with them into that harsh environment that they will need. If the
> >craft is buffeted by periodic winds, they do not have the luxury of calling
> >back to base and saying...
> ...etc...
> Dear "someone", I would strongly recommend that
> you go to Mars at earliest possible time. As for random number
> generation, we sell them for $170...
> http://www.protego.se/sg100_en.htm
>
> Bo Dömstedt
> Protego Information AB

 Gnaediger Herr Bo Doemstedt!
  In all the hype that is there on the WWW page you mentioned,
there is not a single word about the underlying physical phenomena
which produces "high level signal random output".

Without understanding this, I will be reluctant myself, and will
strongly discourage everybody from trusting your (T or P)RNG.
Respectfully BNK

Subject: Re: hardRandNumbGen
Date: Mon, 01 Feb 1999 17:31:14 GMT
From: bo.doemstedt@mbox200.swipnet.se (Bo Dömstedt)
Message-ID: <36b5e46a.951026902@nntpserver.swip.net>
References: <36B48D2D.6B25@erols.com>
Newsgroups: sci.crypt
Lines: 34

"Kazak, Boris" <bkazak@erols.com> wrote:
> Gnaediger Herr Bo Doemstedt!
> In all the hype that is there on the WWW page you mentioned,
>there is not a single word about the underlying physical phenomena
>which produces "high level signal random output".
> Without understanding this, I will be reluctant myself, and will
>strongly discourage everybody from trusting your (T or P)RNG.
> Respectfully BNK

Please excuse us if our server pages are not perfect. We use
a noisy diode as our noise source. It has been extensively
checked, and works well in practice.

handWave <shaken@stirred3.com> wrote:
>And if I buy one, it will be on its own in my environment. You will not
>be there to protect it from my attacks. Mars-like temperatures,
>carbon-dioxide at 1/30 atmosphereic temperature,
>found on your warehouse shelf. It will be subjected to radio beacons at
>close range.
The correct operation of the SG100 device is continuously checked
by a software driver. These attacks will, if successful, result in
an error-message. The SG100 works well even if subject to
strong RF-fields (30V/m).

>Will you be there to help it maintain its insanity, or will you be warm
>and safe in Houston? You will be sipping your coffee while your lonely
>RNG will be under hostile attack by Martians.
Houston is on the other side of the Globe!

Bo Dömstedt
Protego Information AB
Malmo, Sweden
http://www.protego.se/sg100_en.htm

Subject: Re: hardRandNumbGen
Date: Sun, 31 Jan 1999 07:53:56 -1000
From: handWave <shaken@stirred3.com>
Message-ID: <36B498B4.6460@stirred3.com>
References: <36b47461.856793353@nntpserver.swip.net>
Newsgroups: sci.crypt
Lines: 30

Bo Dömstedt wrote:
>
> "someone" wrote:
> >Integrating a random number generator (RNG) on a commodity
> >IC is similar to a manned expedition to MARS: they must take
> >everything with them into that harsh environment that they will need. If the
> >craft is buffeted by periodic winds, they do not have the luxury of calling
> >back to base and saying...
> ...etc...
> Dear "someone", I would strongly recommend that
> you go to Mars at earliest possible time. As for random number
> generation, we sell them for $170...
> http://www.protego.se/sg100_en.htm
>
> Bo Dömstedt
> Protego Information AB

And if I buy one, it will be on its own in my environment. You will not
be there to protect it from my attacks. Mars-like temperatures,
carbon-dioxide at 1/30 atmosphereic temperature, radiation of types not
found on your warehouse shelf. It will be subjected to radio beacons at
close range. If I become deperate, the RNG will be subjected to physical
tampering and disassembly. Will you be there to help it maintain its
insanity, or will you be warm and safe in Houston? You will be sipping
your coffee while your lonely RNG will be under hostile attack by
Martians. The randomness will be compromised, and you will not hear its
screams as it goes from random to silent to Martian Language. Then will
you be so quick to scoff at analogies? No, you will will erect cenotaph
which says: We Loved RNG As If It Were Our Own, Now It Serves Another
Master, Another Purpose.

Terry Ritter, his current address, and his top page.

Last updated: 1999-02-21

The Pentium III RNG

A *Ciphers By Ritter* Page

Most of this discussion concerns the privacy aspects of having a serial number on a processor chip. But there are a few articles about hardware random number generation technology.

Contents

- 1999-01-21 fungus: "Intel has announced that the Pentium III will have a built in hardware random number generator, and individual serial number on each chip."
- 1999-01-20 Brad Aisa: "I don't quite understand how a unique serial number in the chip is supposed to be helpful for anything cryptographic."
- 1999-01-20 Trevor Jackson, III: "The purpose is to *identify* the user, not assist him."
- 1999-01-23 Daniel James: "The serial number in the chip is to help control the trade in stolen CPUs...."
- 1999-01-02 John Savard: "...the serial number will identify the type of the chip - and its rated clock speed...."
- 1999-01-25 Darren New: "Anyway, who would be checking for whether the CPUs are stolen?"
- 1999-01-27 Gurripato (x=nospam): "And, naturally, the desire from the FBI,CIA,NSA ..... to control people´s actions... has NOTHING to do with it."
- 1999-01-28 Daniel James: "...every CPU made will have a unique identifier that cannot be file off, painted over or otherwise rendered illegible...."
- 1999-01-28 fungus: "...except that the numbers will follow no pattern, and Intel won't be keeping records of which chips have which numbers (or so they say)."
- 1999-01-28 Brad Templeton: "They will expect people building PCs to probably run a little program that reads the SN, and checks a database to see if the chip is stolen."
- 1999-01-24 William Hugh Murray: "...it must be a terrible temptation to governments."
- 1999-01-02 John Savard: "The serial number on the chip is to assist in copy-protection schemes...."
- 1999-01-21 .´¯`·..´¯`·..´¯`·..´¯`·..´¯`·..´ "...I invented this 13 years ago while I was working at Intel."
- 1999-01-21 David Boreham: "...couldn't they use laser-zapped fuses for the chip ID ?"
- 1999-01-21 .´¯`·..´¯`·..´¯`·..´¯`·..´¯`·..´ "One problem with fuses for cryptographic keys or for security serial numbers is that they are visible under a microscope." "The random number generator I had patented at Intel is probably similar to the one on the Pentium III: several oscillators combine their outputs and that odd waveform is sampled asynchronously."
- 1999-01-22 Terry Ritter: "I think this is generally a bad way to build an on-chip RNG." "There are two issues here: Random noise, which, because it is random, need not be protected; and a complex sequence, which, because it is complex, *seems* random."
- 1999-01-21 fungus: "Any big selling program will probably cause a lot of headaches if people try to key it to the machine."
- 1999-01-21 Mok-Kong Shen: "Licensed software can be downloaded that works only on machines of certain serial numbers."
- 1999-01-21 John Savard: "The serial number may not be used much for copy-protecting mass-market programs, but it will be handy for some programs that are licensed on a lower-volume basis."
- 1999-01-22 frankrubin@my-dejanews.com: "John, the random number generators found in hardware and software today are almost always 32-bit generators...."
- 1999-01-22 Stefan Axelsson: "Huh? What *state*? We're (hopefully) not discussing a *pseudo* random device in hardware here."
- 1999-01-22 R. Knauer:
- 1999-01-22 John Savard: "I'm well aware that typical 'random number generators' are mixed congruential - with 32-bit, or even 16-bit, internal states."
- 1999-01-21 janb@pmatrix.com: "The primary use for a processor serial number seems like it would be to enforce software licenses."
- 1999-01-21 R. Knauer: "Just great! You buy a computer with an installed OS, the processor dies from infant morality, and now you got a major hassle on your hands."
- 1999-01-22 janb@pmatrix.com: "...long key's are good. None of the enhancements Intel has announced will have any effect on the security of your data." "I've written about five generations of cryptographic random number generators for assorted applications on Intel machines."
- 1999-01-22 R. Knauer: "To date no one has come up with a proveably secure method other than a hardware TRNG...."
- 1999-01-22 janb@pmatrix.com: "The three algorithms included: 1) clock skew bit generator... 2) sound card noise collection... 3) time variations in physical disk seeks...."
- 1999-01-23 Terry Ritter: "I dispute the idea that this is particularly 'random.' What it is is 'complex,' but this is the complexity of digital logic state machines whose structure is known."
- 1999-01-23 janb@pmatrix.com: "Do you also believe disk seeks are highly predicatble?"
- 1999-01-24 Terry Ritter: "I do *indeed* believe that disk seeks are highly predictable...."
- 1999-02-1 Stefek Zaba: "...at a plausible guess - well over 90% of "cryptographic random numbers" in fielded use today are being generated on web servers and clients."
- 1999-01-22 wtshaw: "Embedded id's are just another step in an endless series of trumphs and overtrumphs to put security on yours or someone else's terms."
- 1999-01-21 Anthony Naggs: "IMO The primary use for a unique processor serial number is for easy identification of stolen processors"
- 1999-01-24 William Hugh Murray: "How about as a quid quo pro for an export license for the RNG?"
- 1999-01-26 Myself: "What do we do about multiprocessor machines?"
- 1999-01-26 Marty Levy: "Does anyone know the mechanism Intel plans to use to put the infamous serial numbers on Pentium III chips?"
- 1999-01-27 handWave: "When I worked at Intel I showed them how to make EPROM cells using the ordinary microprocessor wafer fabrication process."
- 1999-01-28 fungus: "...so you're personally to blame for all this!"
- 1999-01-27 Bruce Barnett: "And what do you do with a 2,4,8 or 64-CPU server?"
- 1999-01-28 janb@pmatrix.com: "My preference would be to have that system identity survive cpu and hardware upgrades/changes."
- 1999-02-03 Bruce Barnett: "Exactly. But putting the ID on the chip makes this difficult/ impossible."
- 1999-01-22 Guy Dawson: "This makes it much easier (well, possible) to determine if a chip is one of a batch of stolen chips."
- 1999-01-22 R. Knauer:
- 1999-01-23 Robert Yoder: "'The new identification number is not targeted against processor remarking...'"
- 1999-01-24 Matthew Skala: "I want to build large parallel processors; if each CPU has its own ID number, that may make it easier for then to talk to each other without interferance."
- 1999-01-24 Robert Yoder: "...I _DO_ have experience with multi-CPU SPARC machines, and in that environment, every CPU has it's own ID based on _WHERE_ it is plugged in, and not on a CPU-specific hard-coded number."
- 1999-01-25 Robert Yoder: "Two different organizations are organizing a boycott of Intel because the CPU ID is 'an invasion of privacy'."

```
Subject: Pentium III...
Date: Thu, 21 Jan 1999 01:37:55 +0100
From: fungus &LT;spam@egg.chips.and.spam.com>
Message-ID: &LT;36A676E3.E8D27B14@egg.chips.and.spam.com>
Newsgroups: sci.crypt
Lines: 13
```

Intel has announced that the Pentium III will have a built in hardware
random number generator, and individual serial number on each chip.

http://www.techweb.com/wire/story/TWB19990120S0017

--
<___/>
/ O O \
_____/ FTB.

Subject: Re: Pentium III...
Date: Wed, 20 Jan 1999 22:39:30 -0500
From: Brad Aisa <baisa@istar.ca>
Message-ID: <36A6A172.8BA38F98@istar.ca>
References: <36A676E3.E8D27B14@egg.chips.and.spam.com>
Newsgroups: sci.crypt
Lines: 14

fungus wrote:
>
> Intel has announced that the Pentium III will have a built in hardware
> random number generator, and individual serial number on each chip.

I don't quite understand how a unique serial number in the chip is
supposed to be helpful for anything cryptographic.

...and if the chip dies?

...and if you switch between computers?

__
Brad Aisa

Subject: Re: Pentium III...
Date: Wed, 20 Jan 1999 22:46:41 -0500
From: "Trevor Jackson, III" <fullmoon@aspi.net>
Message-ID: <36A6A320.DDDB9C5F@aspi.net>
References: <36A6A172.8BA38F98@istar.ca>
Newsgroups: sci.crypt
Lines: 21

Read the fine print. The purpose is to *identify* the user, not assist him.

Brad Aisa wrote:

> fungus wrote:
> >
> > Intel has announced that the Pentium III will have a built in hardware
> > random number generator, and individual serial number on each chip.
>
> I don't quite understand how a unique serial number in the chip is
> supposed to be helpful for anything cryptographic.
>
> ...and if the chip dies?
>
> ...and if you switch between computers?
>
> __
> Brad Aisa

Subject: Re: Pentium III...
Date: Sat, 23 Jan 1999 18:50:52 GMT
From: Daniel James <internet@nospam.demon.co.uk>
Message-ID: <VA.00000154.06396550@barney.sonadata>
References: <36A6A320.DDDB9C5F@aspi.net>
Newsgroups: sci.crypt
Lines: 12

In article <36A6A320.DDDB9C5F@aspi.net>, Trevor Jackson, III wrote:
> Read the fine print. The purpose is to *identify* the user, not assist him.
>

The serial number in the chip is to help control the trade in stolen CPUs,
which is a big moneyspinner in certain parts of the criminal world.

Cheers,
 Daniel James
 Daniel at sonadata.demon.co.uk

Subject: Re: Pentium III...
Date: 23 Jan 99 22:26:15 GMT

From: jsavard@ecn.ab.ca ()
Message-ID: <36aa4c87.0@ecn.ab.ca>
References: <VA.00000154.06396550@barney.sonadata>
Newsgroups: sci.crypt
Lines: 17

Daniel James (internet@nospam.demon.co.uk) wrote:
: The serial number in the chip is to help control the trade in stolen CPUs,
: which is a big moneyspinner in certain parts of the criminal world.

I hadn't thought of that, but you are correct.

Also, as the serial number will identify the type of the chip - and its rated clock speed - that should help combat *fraudulent* overclocking in a way that does not require Intel to design the chips to prevent overclocking by the individual user.

However, a serial number accessible to software will be used by some software packages for software piracy prevention: it will not be suitable for most mass-market software, but there are packages to which that sort of thing is applicable.

John Savard

Subject: Re: Pentium III...
Date: Mon, 25 Jan 1999 17:59:10 GMT
From: Darren New <dnew@messagemedia.com>
Message-ID: <36ACB1A7.8640A313@messagemedia.com>
References: <36aa58fe.19369792@nntp.ix.netcom.com>
<VA.00000154.06396550@barney.sonadata>
Newsgroups: sci.crypt
Lines: 20

> >The serial number in the chip is to help control the trade in stolen CPUs,
> >which is a big moneyspinner in certain parts of the criminal world.
>
> Once again the law-abiding citizen has to pay the price for the
> ineptness of law enforcement.

I find this amusing, coming from the newsgroup with likely the most vocal opponents to key escrow. :-)

Anyway, who would be checking for whether the CPUs are stolen? Will Intel refuse to sell you chips unless you promise to check that every chip you buy is not on the hotlist? And require you to sign same with all the people you redistribute to? If I wind up with a stolen chip in my machine, can it be confiscated as stolen property? Sheesh.

--
Darren New / Senior Software Architect / MessageMedia, Inc.
San Diego, CA, USA (PST). Cryptokeys on demand.
"You could even do it in C++, though that should only be done
by folks who think that self-flagellation is for the effete."

Subject: Re: Pentium III...
Date: Wed, 27 Jan 1999 08:13:07 GMT
From: aquiranx@goliatx.ugr.es (Gurripato (x=nospam))
Message-ID: <36aec9e6.2293499@news.cica.es>
References: <36ACB1A7.8640A313@messagemedia.com>
Newsgroups: sci.crypt
Lines: 17

On Mon, 25 Jan 1999 17:59:10 GMT, Darren New <dnew@messagemedia.com>
wrote:

>> >The serial number in the chip is to help control the trade in stolen CPUs,
>> >which is a big moneyspinner in certain parts of the criminal world.

And, naturally, the desire from the FBI,CIA,NSA ..... (your favorite 3-letter agency goes here) to control people´s actions and movements, regardless of whether you belong to the bad guys or not, has NOTHING to do with it.

>> Once again the law-abiding citizen has to pay the price for the
>> ineptness of law enforcement.

I would rather say data-greed. They are far from inept.

Subject: Re: Pentium III...
Date: Thu, 28 Jan 1999 10:28:54 GMT
From: Daniel James <internet@nospam.demon.co.uk>
Message-ID: <VA.0000015c.058b856b@barney.sonadata>
References: <36ACB1A7.8640A313@messagemedia.com>
Newsgroups: sci.crypt
Lines: 15

In article <36ACB1A7.8640A313@messagemedia.com>, Darren New wrote:
> Anyway, who would be checking for whether the CPUs are stolen?

>

The point is that every CPU made will have a unique identifier that
cannot be file off, painted over or otherwise rendered illegible
without destroying the CPU. This will be useful, for example, when the
police find a A.Felon Esq. in posession of a shedful of used CPUs; it
will be possible to verify that they were stolen and from whom.

Cheers,
 Daniel James
 Daniel at sonadata.demon.co.uk

Subject: Re: Pentium III...
Date: Thu, 28 Jan 1999 17:31:35 +0100
From: fungus <spam@egg.chips.and.spam.com>
Message-ID: <36B090E7.EF5B55AD@egg.chips.and.spam.com>
References: <36A676E3.E8D27B14@egg.chips.and.spam.com>
Newsgroups: sci.crypt
Lines: 23

Daniel James wrote:
>
> This will be useful, for example, when the
> police find a A.Felon Esq. in posession of a shedful of used CPUs; it
> will be possible to verify that they were stolen and from whom.
>

...except that the numbers will follow no pattern, and Intel
won't be keeping records of which chips have which numbers
(or so they say).

A distributor could, in theory, take every single chip out of the
box and record all the serial numbers before he puts them in a truck
for transportation. I personally don't think this is very likely....

--
<___/>
/ O O \
_____/ FTB.

Subject: Re: Pentium III...
Date: 28 Jan 1999 12:04:16 PST
From: bt@templetons.com (Brad Templeton)
Message-ID: <78qfs0$c25@journal.concentric.net>
References: <36B090E7.EF5B55AD@egg.chips.and.spam.com>
Newsgroups: sci.crypt
Lines: 34

In article <36B090E7.EF5B55AD@egg.chips.and.spam.com>,
fungus <spam@egg.chips.and.spam.com> wrote:
>
>
>Daniel James wrote:
>>
>> This will be useful, for example, when the
>> police find a A.Felon Esq. in posession of a shedful of used CPUs; it
>> will be possible to verify that they were stolen and from whom.
>>
>
>...except that the numbers will follow no pattern, and Intel
>won't be keeping records of which chips have which numbers
>(or so they say).
>
>A distributor could, in theory, take every single chip out of the
>box and record all the serial numbers before he puts them in a truck
>for transportation. I personally don't think this is very likely....

They won't record what individual has what serial number, but you can
bet they will record what distributors and PC vendors have what serial
numbers.

They will expect people building PCs to probably run a little program
that reads the SN, and checks a database to see if the chip is stolen.
Any legit builder of PCs buying chips from a 3rd party may take a random
sample and test it before paying.

Intel could also put up a web site where customers could check if their
chip is stolen, with the provision that if it is, Intel will give them
the chip (ie. make it un-stolen) so long as they say who they bought it
from. Plus perhaps some other reward.
--
        Brad Templeton   http://www.templetons.com/brad/

Subject: Re: Pentium III...
Date: Sun, 24 Jan 1999 18:33:11 GMT

From: William Hugh Murray <whmurray@sprynet.com>
Message-ID: <36AB6766.E0D9245F@sprynet.com>
References: <36a765d4.0@nnrp1.news.uk.psi.net>
<36A6A172.8BA38F98@istar.ca>
Newsgroups: sci.crypt
Lines: 22

No it shouldn't; particularly since Intel provides the user a control
over it. Still, it must be a terrible temptation to governments.

burt wrote:
>
> Shouldnt be to dificult to get around the serial number..
>
> Brad Aisa wrote in message <36A6A172.8BA38F98@istar.ca>...
> >fungus wrote:
> >>
> >> Intel has announced that the Pentium III will have a built in hardware
> >> random number generator, and individual serial number on each chip.
> >
> >I don't quite understand how a unique serial number in the chip is
> >supposed to be helpful for anything cryptographic.
> >
> >...and if the chip dies?
> >
> >...and if you switch between computers?
> >
> >__
> >Brad Aisa

Subject: Re: Pentium III...
Date: 21 Jan 99 04:59:39 GMT
From: jsavard@ecn.ab.ca ()
Message-ID: <36a6b43b.0@ecn.ab.ca>
References: <36A676E3.E8D27B14@egg.chips.and.spam.com>
Newsgroups: sci.crypt
Lines: 18

fungus (spam@egg.chips.and.spam.com) wrote:

: Intel has announced that the Pentium III will have a built in hardware
: random number generator, and individual serial number on each chip.

: http://www.techweb.com/wire/story/TWB19990120S0017

Hmm. The serial number on the chip is to assist in copy-protection
schemes, creating a market for cryptographic techniques...

and a hardware random number generator on the chip will be useful to
cryptography programs.

So useful, I'm surprised they included such a feature (yes, I know dice
aren't export controlled) since they probably have enough headaches
getting approval to export their latest and greatest microprocessors.

John Savard

Subject: Re: Pentium III...
Date: Thu, 21 Jan 1999 01:11:57 -1000
From: ".·´¯`···´¯`···´¯`···´¯`···´¯`···´¯`·." <untell@outside.com>
Message-ID: <36A70B7D.78BD@outside.com>
References: <36a6b43b.0@ecn.ab.ca>
Newsgroups: sci.crypt
Lines: 37

jsavard@ecn.ab.ca wrote:
>
> fungus (spam@egg.chips.and.spam.com) wrote:
>
> : Intel has announced that the Pentium III will have a built in hardware
> : random number generator, and individual serial number on each chip.
>
> : http://www.techweb.com/wire/story/TWB19990120S0017
>
> Hmm. The serial number on the chip is to assist in copy-protection
> schemes, creating a market for cryptographic techniques...
>
> and a hardware random number generator on the chip will be useful to
> cryptography programs.
>
> So useful, I'm surprised they included such a feature (yes, I know dice
> aren't export controlled) since they probably have enough headaches
> getting approval to export their latest and greatest microprocessors.
>
> John Savard

I spoke to Intel's David Aucsmith about this. Last yesr. I informed him
that I invented this 13 years ago while I was working at Intel. I
invented a random number generator for Intel and I proposed the
non-volatile memory scheme for holding chip serial numbers. I invented
the "single-poly EPROM cell" for storing ID numbers, etc. using the

standard single poly microprocessor wafer process! They said they could not use it because the cells were so big. The n+ diffusion and n-well form one plate of a capacitor, poly 1 forms the other plate, and the floqqting gate. I drew the memory cell, put in on a test mask set, got the wafers, tested the cells, they worked! I wonder if they patented my inventions without notifying me. I might call them soon to collect some royalties. In 1984 I told Larry Palley and Kurt Robinson of Intel Folsom about it. My patent notebook is in the desk of Greg Ledenbach in Folsom California. I have written proof.

Name witheld (for now).

Subject: Re: Pentium III...
Date: Thu, 21 Jan 1999 08:54:32 -0800
From: David Boreham <dboreham@netscape.com>
Message-ID: <36A75BC8.3583A332@netscape.com>
References: <36A70B7D.78BD@outside.com>
Newsgroups: sci.crypt
Lines: 12

> I spoke to Intel's David Aucsmith about this. Last yesr. I informed him
> that I invented this 13 years ago while I was working at Intel. I
> invented a random number generator for Intel and I proposed the
> non-volatile memory scheme for holding chip serial numbers. I invented
> the "single-poly EPROM cell" for storing ID numbers, etc. using the

Pretty cool. But couldn't they use laser-zapped fuses
for the chip ID ? EPROM technology is certainly capable
of interesting things, but for the task at hand
(configure a unique ID at backend test time), you
don't _need_ an EPROM cell.

Subject: Re: Pentium III...
Date: Thu, 21 Jan 1999 19:03:56 -1000
From: "..´¯`...´¯`...´¯`...´¯`...´¯`...´¯`.." <old1@wave.com>
Message-ID: <36A806BC.23A2@wave.com>
References: <36A75BC8.3583A332@netscape.com>
Newsgroups: sci.crypt
Lines: 57

David Boreham wrote:
>
> > I spoke to Intel about this last year. I informed them
> > that I invented this 13 years ago while I was working at Intel. I
> > invented a random number generator for Intel and I proposed the
> > non-volatile memory scheme for holding chip serial numbers. I invented
> > the "single-poly EPROM cell" for storing ID numbers, etc. using the

> Pretty cool. But couldn't they use laser-zapped fuses
> for the chip ID ? EPROM technology is certainly capable
> of interesting things, but for the task at hand
> (configure a unique ID at backend test time), you
> don't _need_ an EPROM cell.

Laser blown fuses are a possibility. I have not seen the
Pentium III chip yet, so the speculations I am making are
only based on years of experience at Intel and 12 years out
of there. One problem with fuses for cryptographic keys or
for security serial numbers is that they are visible under
a microscope. A second problem is that they cause reliability
problems if extra processing steps are not taken: the laser
blast cracks open the top oxide layers and lets contaminations
in. That is why I told Intel to use the single poly EPROM cell
in 1986 for serial numbers on the 80386. I am glad they
finally have taken my advice (or maybe they re-invented the
idea).

The random number generator I had patented at Intel is probably
similar to the one on the Pentium III: several oscillators
combine their outputs and that odd waveform is sampled
asynchronously. Switched capacitors change the loading on each
oscillator with the switches controlled by the random bits that
are accumulated in a shift register. The oscillators each have
different responses to power supply noise, temperature,
capacitance change, noise due to thermally induced voltage
irregularities, and small processing irregularities. A heater
near one oscillator is switched on and off to produce a thermal
history that is different from that of the other oscillators.

When the chip comes out, people who evaluate it should look for
the single poly EPROM cells that are shaped like ping-pong
paddles made of polycrystalline silicon. The handle is the gate
of an MOS transistor, the paddle is the capacitor that stores
the floating charge. High voltages are applied during wafer
sort to this poly capacitor to cause avalanche injection to
program the serial numbers through a bond pad that is not
connected to the package the customer uses. It will be placed
under a metal power bus for added security.

The cells also store wafer ID info so that statistical analysis

can be done and failure tracking can be done years after retail
sales. This would facilitate recalls of chips from bad fab runs.

That's my guess.

An Un-named Source

Subject: Re: Pentium III...
Date: Fri, 22 Jan 1999 19:38:00 GMT
From: ritter@io.com (Terry Ritter)
Message-ID: <36a8d388.8625558@news.io.com>
References: <36A806BC.23A2@wave.com>
Newsgroups: sci.crypt
Lines: 39

On Thu, 21 Jan 1999 19:03:56 -1000, in <36A806BC.23A2@wave.com>, in
sci.crypt ".·´¯`···´¯`···´¯`···´¯`···´¯`···´¯`·." <old1@wave.com>
wrote:

>[...]
>The random number generator I had patented at Intel is probably
>similar to the one on the Pentium III: several oscillators
>combine their outputs and that odd waveform is sampled
>asynchronously. Switched capacitors change the loading on each
>oscillator with the switches controlled by the random bits that
>are accumulated in a shift register. The oscillators each have
>different responses to power supply noise, temperature,
>capacitance change, noise due to thermally induced voltage
>irregularities, and small processing irregularities. A heater
>near one oscillator is switched on and off to produce a thermal
>history that is different from that of the other oscillators.

I think this is generally a bad way to build an on-chip RNG.

There are two issues here: Random noise, which, because it is random,
need not be protected; and a complex sequence, which, because it is
complex, *seems* random. These are two very different things. If we
have a machine which only does both, it will be very difficult to know
just how much noise we really have. And if there is not much noise
there, we have yet another cipher or pseudorandom RNG (admittedly of a
special form, with special features, like thermally-delayed bit
correlations) which might be attacked.

I think it is generally inappropriate to try to hide flaws in the
random source. We can still use flawed randomness, we just have to
process it. And if we can characterize the flaws, we can get an idea
about how much we have to process.

Terry Ritter ritter@io.com http://www.io.com/~ritter/
Crypto Glossary http://www.io.com/~ritter/GLOSSARY.HTM

Subject: Re: Pentium III...
Date: Thu, 21 Jan 1999 13:18:47 +0100
From: fungus <spam@egg.chips.and.spam.com>
Message-ID: <36A71B27.6CE10C77@egg.chips.and.spam.com>
References: <36a6b43b.0@ecn.ab.ca>
Newsgroups: sci.crypt
Lines: 47

jsavard@ecn.ab.ca wrote:
>
> fungus (spam@egg.chips.and.spam.com) wrote:
>
> : Intel has announced that the Pentium III will have a built in hardware
> : random number generator, and individual serial number on each chip.
>
> : http://www.techweb.com/wire/story/TWB19990120S0017
>
> Hmm. The serial number on the chip is to assist in copy-protection
> schemes, creating a market for cryptographic techniques...
>

Possibly. Most Unix machines have had serial numbers for years now
and software is usage is often keyed to the machine. I'm not sure how
it will work in the mass market of the PC world though. Any big selling
program will probably cause a lot of headaches if people try to key
it to the machine.

> and a hardware random number generator on the chip will be useful to
> cryptography programs.
>

Maybe not as useful as people will think. It can be used for session
keys etc., but I doubt if people will use it for OTP (cue new thread!)
due to all the key management problems that involves. The question is
whether a hardware number generator provides much benefit over a software

generator for session keys.

> So useful, I'm surprised they included such a feature (yes, I know dice
> aren't export controlled) since they probably have enough headaches
> getting approval to export their latest and greatest microprocessors.
>

Do random number generators fall under export restrictions? I would
say "only if they can be seeded and synched with other machines". If
this isn't the case then I don't see how they can have problems. I'm
sure a company like Intel has thought this through pretty carefully....

--
<___/>
/ O O \
_____/ FTB.

Subject: Re: Pentium III...
Date: Thu, 21 Jan 1999 17:00:35 +0100
From: Mok-Kong Shen <mok-kong.shen@stud.uni-muenchen.de>
Message-ID: <36A74F23.6BBB7A9C@stud.uni-muenchen.de>
References: <36A71B27.6CE10C77@egg.chips.and.spam.com>
Newsgroups: sci.crypt
Lines: 35

fungus wrote:
>
> jsavard@ecn.ab.ca wrote:
> >
> > fungus (spam@egg.chips.and.spam.com) wrote:

> > Hmm. The serial number on the chip is to assist in copy-protection
> > schemes, creating a market for cryptographic techniques...
> >
>
> Possibly. Most Unix machines have had serial numbers for years now
> and software is usage is often keyed to the machine. I'm not sure how
> it will work in the mass market of the PC world though. Any big selling
> program will probably cause a lot of headaches if people try to key
> it to the machine.
>
> > and a hardware random number generator on the chip will be useful to
> > cryptography programs.

Licensed software can be downloaded that works only on machines
of certain serial numbers. I don't see a difference here between
UNIX workstantions and PC.

>
> Maybe not as useful as people will think. It can be used for session
> keys etc., but I doubt if people will use it for OTP (cue new thread!)
> due to all the key management problems that involves. The question is
> whether a hardware number generator provides much benefit over a software
> generator for session keys.

Presumably a good session key can be constructed from a combination
of the generator output and input from the user. This would eliminate
questions concerning the quality of the hardware generator.

M. K. Shen

Subject: Re: Pentium III...
Date: Thu, 21 Jan 1999 18:18:28 GMT
From: jsavard@tenMAPSONeerf.edmonton.ab.ca (John Savard)
Message-ID: <36a76ea8.3406965@news.prosurfr.com>
References: <36A71B27.6CE10C77@egg.chips.and.spam.com>
Newsgroups: sci.crypt
Lines: 48

fungus <spam@egg.chips.and.spam.com> wrote, in part:

>Possibly. Most Unix machines have had serial numbers for years now
>and software is usage is often keyed to the machine. I'm not sure how
>it will work in the mass market of the PC world though. Any big selling
>program will probably cause a lot of headaches if people try to key
>it to the machine.

The serial number may not be used much for copy-protecting mass-market
programs, but it will be handy for some programs that are licensed on
a lower-volume basis.

As a Pentium computer is comparable to the mainframes of yesteryear,
some systems with server-type hardware are used for quite impressive
applications.

(Of course, the big news about the Pentium III seems to be that it
includes the long-awaited MMX 2 instruction set.)

>Maybe not as useful as people will think. It can be used for session
>keys etc., but I doubt if people will use it for OTP (cue new thread!)

>due to all the key management problems that involves. The question is
>whether a hardware number generator provides much benefit over a software
>generator for session keys.

>Do random number generators fall under export restrictions? I would
>say "only if they can be seeded and synched with other machines". If
>this isn't the case then I don't see how they can have problems. I'm
>sure a company like Intel has thought this through pretty carefully....

My understanding is that the chip will include hardware to generate
true random numbers from electrical noise.

This is very helpful to encryption programs, since it allows session
keys to be as secure as the public-key method used to transmit them;
getting true randomness from keypresses, disk movement, and so on, is
cumbersome and depends on the individual user's configuration, and not
using truly random numbers creates an additional point of weakness in
a cryptographic system.

Export restrictions on hardware are generally more stringent than on
software, and the Pentium III chip could even provoke the enacting of
new restrictions that don't exist at present. (Of course, new chips
that push the envelope of performance aren't exportable for a while
after introduction in any case.)

John Savard
http://www.freenet.edmonton.ab.ca/~jsavard/index.html

Subject: Re: Pentium III...
Date: Fri, 22 Jan 1999 08:48:54 GMT
From: frankrubin@my-dejanews.com
Message-ID: <789e1kbfi1@nnrp1.dejanews.com>
References: <36a6b43b.0@ecn.ab.ca>
Newsgroups: sci.crypt
Lines: 27

In article <36a6b43b.0@ecn.ab.ca>,
  jsavard@ecn.ab.ca () wrote:

>
> Hmm. The serial number on the chip is to assist in copy-protection
> schemes, creating a market for cryptographic techniques...
> and a hardware random number generator on the chip will be useful to
> cryptography programs.
>
> So useful, I'm surprised they included such a feature (yes, I know dice
> aren't export controlled) since they probably have enough headaches
> getting approval to export their latest and greatest microprocessors.
>
> John Savard
>

John, the random number generators found in hardware and software today are
almost always 32-bit generators, meaning they have only 32-bit internal
states. This is equivalent to a 32-bit crypto key, which is far too small to
make a secure cryptographic system. Nobody will worry about exporting such a
chip.

If you are looking for something more secure, see my article "One-Time Pad
Cryptography" in the Oct. 1996 Cryptologia.

----------== Posted via Deja News, The Discussion Network ==----------
http://www.dejanews.com/ Search, Read, Discuss, or Start Your Own

---

Subject: Re: Pentium III...
Date: 22 Jan 1999 13:34:21 GMT
From: sax@rmovt.rply.ce.chalmers.se (Stefan Axelsson)
Message-ID: <789uot$5ag$1@nyheter.chalmers.se>
References: <789e1kbfi1@nnrp1.dejanews.com>
Newsgroups: sci.crypt
Lines: 16

In article <789e1kbfi1@nnrp1.dejanews.com>,
 <frankrubin@my-dejanews.com> wrote:

>John, the random number generators found in hardware and software today are
>almost always 32-bit generators, meaning they have only 32-bit internal
>states.

Huh? What *state*? We're (hopefully) not discussing a *pseudo* random
device in hardware here. The question of whether it is useful, or any
good, still remains to be answered, of course.

Stefan,
--
Stefan Axelsson Chalmers University of Technology
sax@rmovt.rply.ce.chalmers.se Dept. of Computer Engineering
(Remove "rmovt.rply" to send mail.)

Subject: Re: Pentium III...

Date: Fri, 22 Jan 1999 14:21:13 GMT
From: rcktexas@ix.netcom.com (R. Knauer)
Message-ID: <36a88936.14838306@nntp.ix.netcom.com>
References: <789e1kbfi1@nnrp1.dejanews.com>
Newsgroups: sci.crypt
Lines: 14

On Fri, 22 Jan 1999 08:48:54 GMT, frankrubin@my-dejanews.com wrote:

>If you are looking for something more secure, see my article "One-Time Pad
>Cryptography" in the Oct. 1996 Cryptologia.

Is that posted to the web? If so, could you give us the link.

Bob Knauer

"It is not the function of our government to keep the citizen from
falling into error; it is the function of the citizen to keep the
government from falling into error."
--Justice Robert H. Jackson

Subject: Re: Pentium III...
Date: Fri, 22 Jan 1999 18:15:32 GMT
From: jsavard@tenMAPSONeerf.edmonton.ab.ca (John Savard)
Message-ID: <36a8be80.6108402@news.prosurfr.com>
References: <789e1kbfi1@nnrp1.dejanews.com>
Newsgroups: sci.crypt
Lines: 31

frankrubin@my-dejanews.com wrote, in part:

>John, the random number generators found in hardware and software today are
>almost always 32-bit generators, meaning they have only 32-bit internal
>states. This is equivalent to a 32-bit crypto key, which is far too small to
>make a secure cryptographic system. Nobody will worry about exporting such a
>chip.

>If you are looking for something more secure, see my article "One-Time Pad
>Cryptography" in the Oct. 1996 Cryptologia.

I'm sure I've seen the article, even if I don't remember it offhand.
Could it be the one where the claim that the one-time-pad is
"impractical" is countered?

I'm well aware that typical 'random number generators' are mixed
congruential - with 32-bit, or even 16-bit, internal states.

However, unless I am very much mistaken, the reference to a 'hardware
random number generator' on the Pentium III does *not* refer to
anything of that type, but to a built-in source of true randomness, by
means of electrical noise or the like. It is that which is at the
present awkward and expensive to add to a PC, while a PRNG is easy to
write in software (and Windows even provides a shared PRNG with a
global seed, seeded by the clock at startup, to approximate true
random behavior - in a fashion adequate, say, for games, if not for
cryptography), and thus it is that which remedies a fundamental
omission.

John Savard
http://www.freenet.edmonton.ab.ca/~jsavard/index.html

Subject: Re: Pentium III...
Date: Thu, 21 Jan 1999 19:37:32 GMT
From: janb@pmatrix.com
Message-ID: <787vlr$3b4$1@nnrp1.dejanews.com>
References: <36A676E3.E8D27B14@egg.chips.and.spam.com>
Newsgroups: sci.crypt
Lines: 30

> Intel has announced that the Pentium III will have a built in hardware
> random number generator, and individual serial number on each chip.

I can think of cases where having a machine serial number would be somewhat
handy. I'm not sure I would build it into the processor though. This
tremendously complicates upgrading/replacing your hardware. I think one of the
already existing serial number chips (like from Dallas Semi, mabey an iButton)
mounted on the motherboard at a well defined address would have been better.

Or even much better, some sort of hardware MD5/SHA chip. If the serial number
is directly readable, it can be faked. If you can only get the effect the
serial number has on some algorithm, like as a MD5 seed, it's a lot harder to
forge. You never would let the actual serial number be known in the clear.

As for putting a random number generator in the processor, that seems just
silly. It's pretty easy to generate extreemly high quality random numbers on a
typical PC at rates of 1000 bits/second. This is way more than sufficent for
most uses.

The primary use for a processor serial number seems like it would be to
enforce software licenses. If I read the Microsoft OS license correctly, your

NOT allowed to move a copy of the OS from an old machine to a new machine.
The OS is licensed to a specific machine, which might be interpreted as a
specific processor.

- Jan

----------== Posted via Deja News, The Discussion Network ==----------
http://www.dejanews.com/ Search, Read, Discuss, or Start Your Own

---

Subject: Re: Pentium III...
Date: Thu, 21 Jan 1999 20:05:07 GMT
From: rcktexas@ix.netcom.com (R. Knauer)
Message-ID: <36a78709.23154884@nntp.ix.netcom.com>
References: <787vlr$3b4$1@nnrp1.dejanews.com>
Newsgroups: sci.crypt
Lines: 36

On Thu, 21 Jan 1999 19:37:32 GMT, janb@pmatrix.com wrote:

>It's pretty easy to generate extreemly high quality random numbers on a
>typical PC at rates of 1000 bits/second.

How is that be done such that no one can crack the resulting cipher
given sufficient resources?

For example, if you have a laptop with sensitive business material on
it, and a competitor steals it, he may decide that it is worth 1
million dollars to decipher your files.

Can the kind of random number generator you allude to above be
adequate to prevent all but the most concerted attacks, like from the
NSA?

>The primary use for a processor serial number seems like it would be to
>enforce software licenses. If I read the Microsoft OS license correctly, your
>NOT allowed to move a copy of the OS from an old machine to a new machine.
>The OS is licensed to a specific machine, which might be interpreted as a
>specific processor.

Just great! You buy a computer with an installed OS, the processor
dies from infant morality, and now you got a major hassle on your
hands.

You would think the industry learned its lesson from the old days of
closed architectures, key-based S/W, dongle keys, etc. I can just see
Ziff Davis refusing to test anything that is Pentium III based.

Bob Knauer

"A man with his heart in his profession imagines and finds
resources where the worthless and lazy despair."
--Frederic the Great, in instructions to his Generals

Subject: Re: Pentium III...
Date: Fri, 22 Jan 1999 01:15:31 GMT
From: janb@pmatrix.com
Message-ID: <788jf9$li3$1@nnrp1.dejanews.com>
References: <36a78709.23154884@nntp.ix.netcom.com>
Newsgroups: sci.crypt
Lines: 65

In article <36a78709.23154884@nntp.ix.netcom.com>,
  rcktexas@ix.netcom.com wrote:
> On Thu, 21 Jan 1999 19:37:32 GMT, janb@pmatrix.com wrote:
>
> >It's pretty easy to generate extreemly high quality random numbers on a
> >typical PC at rates of 1000 bits/second.
>
> How is that be done such that no one can crack the resulting cipher
> given sufficient resources?

The method of random session key generation has nothing to do with ease of
brute force attacks on a specific cryptographic cipher. Cracking a 56-bit DES
key will take exactly the same computing effort (22 hours on Distributed.net)
no matter what the source of the key.

> For example, if you have a laptop with sensitive business material on
> it, and a competitor steals it, he may decide that it is worth 1
> million dollars to decipher your files.

I agree, long key's are good. None of the enhancements Intel has announced
will have any effect on the security of your data. Adding a thumbprint reader
or smart card reader or Dallas iButton reader (see http://www.ibutton.com)
would.

If anything, I think there is danger of a false sense of security. Just
because some computer has a Pentium III, with a processor serial number and
hardware random number generator doesn't mean it's any more or less secure
than a system with say an AMD K6-2 processor.

> Can the kind of random number generator you allude to above be

> adequate to prevent all but the most concerted attacks, like from the
> NSA?

Yes, I believe it's possible to generate very high quality random numbers, without thermal noise hardware, like Intel is planning to add to the Pentium III. Random number generators are used for key generation.

I have quite a lot of experience in this specific issue. I've written about five generations of cryptographic random number generators for assorted applications on Intel machines. The latest generation I believe makes extreemly high quality random numbers.

> >The primary use for a processor serial number seems like it would be to
> >enforce software licenses. If I read the Microsoft OS license correctly, your
> >NOT allowed to move a copy of the OS from an old machine to a new machine.
> >The OS is licensed to a specific machine, which might be interpreted as a
> >specific processor.
>
> Just great! You buy a computer with an installed OS, the processor
> dies from infant morality, and now you got a major hassle on your
> hands.
>
> You would think the industry learned its lesson from the old days of
> closed architectures, key-based S/W, dongle keys, etc. I can just see
> Ziff Davis refusing to test anything that is Pentium III based.

At least a dongle would work with your replacement processor. Of cource if your dongle breaks (or you loose it) you may be stuck. I like thumbprint or retina scans more and more every day. Or a smart card, with a duplicate in your safe deposit box (SDB).

- Jan

----------== Posted via Deja News, The Discussion Network ==----------
http://www.dejanews.com/ Search, Read, Discuss, or Start Your Own

---

Subject: Re: Pentium III...
Date: Fri, 22 Jan 1999 14:18:11 GMT
From: rcktexas@ix.netcom.com (R. Knauer)
Message-ID: <36a8840e.13518158@nntp.ix.netcom.com>
References: <788jf9$li3$1@nnrp1.dejanews.com>
Newsgroups: sci.crypt
Lines: 59

On Fri, 22 Jan 1999 01:15:31 GMT, janb@pmatrix.com wrote:

>Yes, I believe it's possible to generate very high quality random numbers,
>without thermal noise hardware, like Intel is planning to add to the Pentium
>III. Random number generators are used for key generation.

That very well may be true if you do not require crypto-grade random numbers.

>I have quite a lot of experience in this specific issue. I've written about
>five generations of cryptographic random number generators for assorted
>applications on Intel machines. The latest generation I believe makes
>extreemly high quality random numbers.

Please elaborate.

Also you might want to join the thread on sci.crypt entitled "Metaphysics of Randomness" where we are discussing the fundamentals of crypto-grade random number generation in terms of considerations such as Kolgomorov-Chaitin complexity theory, Godel's Theorem and Turing's Halting Problem, decorrelation schemes for text ciphers, digit expansion generators for irrational numbers and transcendentals and other schemes to generate random numbers.

To date no one has come up with a proveably secure method other than a hardware TRNG - although some have claimed their methods are practically secure to a very close level of approximation. The criterion is to produce an OTP cipher system which can withstand a Bayesian attack, yet not require distribution of the pads.

>At least a dongle would work with your replacement processor. Of cource if
>your dongle breaks (or you loose it) you may be stuck.

I cracked a dongle once - it is not all that difficult if you trap the strings it expects and then write a wedge to supply them.

>I like thumbprint or
>retina scans more and more every day. Or a smart card, with a duplicate in
>your safe deposit box (SDB).

I guess my problem is that there is no real need for all this. The apparent money lost in pirated software is far less than the money gained when people buy the S/W later, especially when you take into account that the people who pirate S/W can't or won't buy it if they have to until they can afford it.

This obsession with sticking one's nose into every nook and cranny of consumer activities is fueled by companies who profit from promoting the technology to do it. Then at a later date it is discovered that

the intrusions were worthless, and in many cases even
counterproductive, in commercial terms.

Bob Knauer

"It is not the function of our government to keep the citizen from
falling into error; it is the function of the citizen to keep the
government from falling into error."
--Justice Robert H. Jackson

Subject: Re: Pentium III...
Date: Fri, 22 Jan 1999 20:19:47 GMT
From: janb@pmatrix.com
Message-ID: <78amgn$eu7$1@nnrp1.dejanews.com>
References: <36a8840e.13518158@nntp.ix.netcom.com>
Newsgroups: sci.crypt
Lines: 104

In article <36a8840e.13518158@nntp.ix.netcom.com>,
  rcktexas@ix.netcom.com wrote:
> On Fri, 22 Jan 1999 01:15:31 GMT, janb@pmatrix.com wrote:
>
> >Yes, I believe it's possible to generate very high quality random numbers,
> >without thermal noise hardware, like Intel is planning to add to the Pentium
> >III. Random number generators are used for key generation.
>
> That very well may be true if you do not require crypto-grade random
> numbers.
>
> >I have quite a lot of experience in this specific issue. I've written about
> >five generations of cryptographic random number generators for assorted
> >applications on Intel machines. The latest generation I believe makes
> >extreemly high quality random numbers.
>
> Please elaborate.

Earlier generations were for projects that supplied crypto support on a number
of Internet products (like web servers). For interactive apps, collecting bits
from mouse position delta's was used.

The latest generation was used on some credit card processing software. Three
algorithms were available, with randomness analysis done to dynamically
select which algorithm to use. Randomness analysis would also flag that the
RND seemed broken, possibly halting the application.

The three algorithms included:

1) clock skew bit generator

Basically spin the processor incrementing a memory location (at like 200
million increments/sec) and then periodically (from a different clock source)
interrupt the processor and extract a single bit from the count. Repeat this
for as many bits as needed. Production rate as high as 1000 bits/sec can be
achieved. Unless the processor code path, cache hits, bus wait states, etc.
between each bit sample is identical, the bit will not be predictable. Also
the small jitter in the two clock sources will cause the relationship between
them to jitter, causing the sample value to vary. Typical clock crystals are
I believe +/- 25 ppm.

2) sound card noise collection

Collect a bunch of samples from the input (line/mic) of a sound card, and then
stir the bits together with a MD5. Production rate is a bit lower and also no
guarantee of a sound card. Also requires exclusve device use. Still, for a
secure server, requiring a sound card (cheaper is better) is not much of an
expense. Most PC already have sounds card now.

3) time variations in physical disk seeks

Perform a large number of physical disk seeks and measure processor clock
resolution timing using RDTSC (currently about 3 nanosecond resolution). Use
the bottom bit as part of random stream (or stir with MD5). Production rate is
slow (50-100 bits/sec), and makes your system noisy for a bit. For reseeding a
PRNG at periodic intervals this is not an problem.

Combined, these methods allow production of very high quality random numbers.
Nearly every currently existing PC can do these with no new processor. These
assume you retain physical security of the machine, which I think is pretty
realistic at the moment of key generation.

If course all these methods are meaningless if an intruder is allowed access
to modify the software.

The same weakness applies to a RNG in a processor. If the processor did the
whole private key generation, and kept it stashed in the processor (on
eeprom), with interfaces to run the crypto algorithms in the processor
(basically a smart card in your Pentium III), then I could see some improved
security. I don't believe this is what was announced by Intel though.

> To date no one has come up with a proveably secure method other than a
> hardware TRNG - although some have claimed their methods are
> practically secure to a very close level of approximation. The

> criterion is to produce an OTP cipher system which can withstand a
> Bayesian attack, yet not require distribution of the pads.

Some of the methods I described in essense are hardware RNG's. For example the disk seek timing is influenced by micro air currents inside the drive case and also temperature variations.

Things to consider include, is the random number predictable and can it be influenced. PSNR's are totally predictable, if you know the algorithm and and get in sync with the stream (not an easy task). I believe the methods I described above are not very predictable, but without physical security, might be influenced (by the NSA with some very significant effort). Even if they could totally be influenced under lab conditions, this would have no effect on the quality of private keys already generated.

My guess is none of the methods, including a Pentium III RND, are guaranteed to pass Tempest security. I could imagine the RFI from the Pentium III RNG might give away clues about it's value, as would the disk drive firmware. I'm not a Tempest hardware or Pentium III designer, so can't say for sure.

If I were REALLY serious about security, I'd probably want some sort of RFI protected, physically secure, crypto coprocessor with it's own eeprom storage of my private keys. A $25 crypto iButton fits this description better than some new instructions on a Pentium III.

- Jan

----------== Posted via Deja News, The Discussion Network ==----------
http://www.dejanews.com/ Search, Read, Discuss, or Start Your Own

---

Subject: Re: Pentium III...
Date: Sat, 23 Jan 1999 00:15:02 GMT
From: ritter@io.com (Terry Ritter)
Message-ID: <36a91475.8443843@news.io.com>
References: <78amgn$eu7$1@nnrp1.dejanews.com>
Newsgroups: sci.crypt
Lines: 82

On Fri, 22 Jan 1999 20:19:47 GMT, in
<78amgn$eu7$1@nnrp1.dejanews.com>, in sci.crypt janb@pmatrix.com
wrote:

>[...]
>1) clock skew bit generator
>
>Basically spin the processor incrementing a memory location (at like 200
>million increments/sec) and then periodically (from a different clock source)
>interrupt the processor and extract a single bit from the count. Repeat this
>for as many bits as needed. Production rate as high as 1000 bits/sec can be
>achieved. Unless the processor code path, cache hits, bus wait states, etc.
>between each bit sample is identical, the bit will not be predictable. Also
>the small jitter in the two clock sources will cause the relationship between
>them to jitter, causing the sample value to vary. Typical clock crystals are
>I believe +/- 25 ppm.

I think this has taken on the status of a crypto "old wives tale,"
and one might well wonder where it comes from. I have been writing
about it for quite some time (see:

http://www.io.com/~ritter/RAND/NICORAND.HTM

especially

http://www.io.com/~ritter/RAND/92062703.HTM

and

http://www.io.com/~ritter/RAND/92110301.HTM),

and I dispute the idea that this is particularly "random." What it is
is "complex," but this is the complexity of digital logic state
machines whose structure is known.

Computer clock signals are developed from analog crystal oscillators,
specifically *because* they are fairly accurate and do not drift much
over time. Although crystals do have slightly different frequencies,
if we know approximate frequencies, we can develop the precise ratio
by looking at the sampled results. Although crystal oscillators do
drift over time and temperature, they don't drift very much, and they
tend to follow a similar pattern from power-on when they do.

One can handwave about "processor code path, cache hits, bus wait
states," but the "processor code path" is the program, and it seems
quite likely that the program used for random bits once will be used
again. So the cache contents will tend to be the same, as will the
bus wait states. We are talking about essentially error-free digital
systems: There is no reason to expect these things to be "random,"
just complex.

In practice, the major factor which encourages the belief in the
"randomness" of such systems is probably memory refresh, which

interrupts processing periodically and -- on the surface -- unexpectedly. But by "periodically" we typically mean a crystal-controlled precise period between interrupts. We will know the expected period, as well as the computation consequences when the interrupt is taken, and any deviations from this will just help us to further refine the exact internal state of the hardware.

The "jitter" in a crystal oscillator is best modeled as noise in the analog-to-digital conversion -- a bipolar cycle-by-cycle phase difference. The magnitude is tiny and typically normally-distributed, so small values are frequent, but large values are rare. So the probability that this sort of jitter will affect the digital result in any particular cycle is very small. But it *will* show up eventually, and when it does, it will reveal the exact internal state of the oscillator. (That is, that it was close enough to the digital square-wave edge to be affected by the tiny noise variation.)

I claim that these sorts of RNG are best understood as fairly-complex PSEUDO-random generators, with only small amounts of uncertainty beyond their internal hardware state. It would be extremely unwise to hope they could oppose a well-equipped and well-financed attack on their sequence.

Terry Ritter ritter@io.com http://www.io.com/~ritter/
Crypto Glossary http://www.io.com/~ritter/GLOSSARY.HTM

Subject: Re: Pentium III...
Date: Sat, 23 Jan 1999 20:08:45 GMT
From: janb@pmatrix.com
Message-ID: <78da88fco1@nnrp1.dejanews.com>
References: <36a91475.8443843@news.io.com>
Newsgroups: sci.crypt
Lines: 61

In article <36a91475.8443843@news.io.com>,
ritter@io.com (Terry Ritter) wrote:
> and I dispute the idea that this is particularly "random." What it is
> is "complex," but this is the complexity of digital logic state
> machines whose structure is known.

If the clock skew were the ONLY factor, this might not be such a good choice. I used the title of clock skew because people would recognize the implementation strategy. There are MANY other activities affecting the code execute code, and also may access memory via DMA, causing memory access wait state variations. If you have methods to predict the time, within 3 nanoseconds, of every network packet flowing into a device, there are some folks from Cisco who would want to talk to you. As far as I know, the instant in time that someone in the world clicks a link on a web page, is a random event. These events will influence the code path of the sample time on a web server, causing true randomness. For any computer that has a UI, every keystroke or mouse moment causes code path variations, triggered by true random events.

The software generator I mentioned also had three algorithms, to deal with the potential that one (or even two) had a weakness. Stirring a less random soure together with a more random source gives the more random result. Do you also believe disk seeks are highly predicatble?

Your also suggesting you can take the output from a cryptographic hash (assuming we hash the data from the original source generator), and exactly predict the patterns of the input, to analyze the patterns originally generated. I've heard of some success in generating MD5 hashes with specific output bits at specific values, but haven't heard of a total breakdown in the integrity of either MD5 or SHA-1. Are you suggesting cryptographic hashes are reversable? That gives a hash, you can calculate the number and value of input bits?

I agree that if you put a system in a lab under very controlled conditions, you may be able to predict or influence things better. The software I worked on and I believe software running on most peoples systems will not be under these conditions.

The topic of this thread is about the usefulness of a hardware random number generator in a Pentium III. My belief is anybody who is serious about security, will want some external hardware device anyway. So doing security on a PC without extra hardware only applies for 'low' security uses. I believe software generated random numbers are very sufficent for this.

Really, I'd love to see some hardware support for security in PC's. I just don't think Intel's latest features improve things much. I don't believe the weakest link is in the quality of random number generators. I think a socket on motherboards for an iButton would be a lot better (and cost very little). For lowest security uses, just a serial number button with costs a buck. If users wanted better security, a crypto iButton could be popped in for a dozen or two bucks. Of course this would work just as well for Intel, AMD, and Cyrix processors, which may not be Intel's goal.

- Jan

P.S. I really have no connection with Dallas Semi. I just like iButtons more and more when I think about the problems that need solving.

----------== Posted via Deja News, The Discussion Network ==----------
http://www.dejanews.com/ Search, Read, Discuss, or Start Your Own

---

Subject: Re: Pentium III...
Date: Sun, 24 Jan 1999 05:08:43 GMT
From: ritter@io.com (Terry Ritter)
Message-ID: <36aaaad8.5611074@news.io.com>
References: <78da88fco1@nnrp1.dejanews.com>
Newsgroups: sci.crypt
Lines: 161

On Sat, 23 Jan 1999 20:08:45 GMT, in
<78da88fco1@nnrp1.dejanews.com>, in sci.crypt janb@pmatrix.com
wrote:

>In article <36a91475.8443843@news.io.com>,
> ritter@io.com (Terry Ritter) wrote:
>> and I dispute the idea that this is particularly "random." What it is
>> is "complex," but this is the complexity of digital logic state
>> machines whose structure is known.
>
>If the clock skew were the ONLY factor, this might not be such a good choice.
>I used the title of clock skew because people would recognize the
>implementation strategy. There are MANY other activities affecting the code
>path between each sample. For example, I/O activity causes the processor to
>execute code, and also may access memory via DMA, causing memory access wait
>state variations.

But I/O activity is programmed -- presumably we will be re-using the
same program, and can expect similar activity. And DMA activity is
also programmed.

If we have other tasks running, AND those tasks are actually doing
something *while* we take "random" samples, that would be more
complex. But if (as we expect) sampling is frequent compared to
task-changes, most samples will be related and unaffected within the
period of the sampling task, making the supposed increase in
complexity largely illusory.

>If you have methods to predict the time, within 3
>nanoseconds, of every network packet flowing into a device, there are some
>folks from Cisco who would want to talk to you.

So, basically, you recommend generating random numbers while actively
on line and communicating with a network. That may not be the best
approach.

If external communications are the key to randomness, we'd sure better
hope The Opponent is not monitoring that line (to say nothing of
actively *influincing* that timing). Can we be sure?

>As far as I know, the instant
>in time that someone in the world clicks a link on a web page, is a random
>event.

For a random person, for one click, yes.

But when that person makes repeated clicks, no.

>These events will influence the code path of the sample time on a web
>server, causing true randomness.

Producing cryptographic random values on a web server does not sound
like a great idea to me.

>For any computer that has a UI, every
>keystroke or mouse moment causes code path variations, triggered by true
>random events.

With respect to keystrokes, every keystroke is first sampled in the
keyboard by a scanning process. That scanning process occurs
periodically, typically under the control of yet another crystal
oscillator. Key strokes are thus quantized in time, which is hardly
"random."

This problem probably does not apply to a mouse. But for this to
work, we have to be using that mouse while we are actively producing
random numbers, and our failure to do so will affect the quality of
the results.

>The software generator I mentioned also had three algorithms, to deal with the
>potential that one (or even two) had a weakness. Stirring a less random soure
>together with a more random source gives the more random result.

It also means that all the discussion and handwaving about the less
random source is essentially irrelevant.

>Do you also
>believe disk seeks are highly predicatble?

I do *indeed* believe that disk seeks are highly predictable (which is not to say that they can be predicted to the nanosecond, but that they can be approximated to virtually arbitrary precision). If we know the current rotational angle, and the number of tracks to cross, and the characteristics of the particular drive, seek time is highly predictable. And we know the rotational angle from the last seek and read, and the time elapsed. Typically, disk rotation is yet another crystal-controlled entity.

>Your also suggesting you can take the output from a cryptographic hash
>(assuming we hash the data from the original source generator), and exactly
>predict the patterns of the input, to analyze the patterns originally
>generated.

I have suggested no such thing.

But it was my understanding that you were talking about *real* randomness, as opposed to a hashed or encrypted counter. If we are willing to accept the latter, all we need is a simple polynomial counter and a cryptographic hash or cipher, and we can avoid all this hardware-software-randomness stuff.

>I've heard of some success in generating MD5 hashes with specific
>output bits at specific values, but haven't heard of a total breakdown in the
>integrity of either MD5 or SHA-1. Are you suggesting cryptographic hashes are
>reversable? That gives a hash, you can calculate the number and value of input
>bits?
>
>I agree that if you put a system in a lab under very controlled conditions,
>you may be able to predict or influence things better. The software I worked
>on and I believe software running on most peoples systems will not be under
>these conditions.

"Belief" is arguably the major problem in cryptography: Most people who propose cryptographic systems *believe* that nobody could crack their complex design. But upon what reality is such belief based?

>The topic of this thread is about the usefulness of a hardware random number
>generator in a Pentium III. My belief is anybody who is serious about
>security, will want some external hardware device anyway. So doing security
>on a PC without extra hardware only applies for 'low' security uses. I
>believe software generated random numbers are very sufficent for this.

"Software generated random numbers" will be sufficent for exactly as long as it takes someone to efficiently solve the system and produce a cracking routine. When cracking is trivial, the result is NO security at all, not even "low" security. And they will not tell you when they succeed.

>Really, I'd love to see some hardware support for security in PC's. I just
>don't think Intel's latest features improve things much. I don't believe the
>weakest link is in the quality of random number generators. I think a socket
>on motherboards for an iButton would be a lot better (and cost very little).
>For lowest security uses, just a serial number button with costs a buck. If
>users wanted better security, a crypto iButton could be popped in for a dozen
>or two bucks. Of course this would work just as well for Intel, AMD, and
>Cyrix processors, which may not be Intel's goal.

This particular discussion was about the recent announcement of a serial number and RNG on the processor chip. But information from Intel last month involves a BIOS storage chip with hardware public-key type operations. The protected BIOS is more likely to be the source of software security than the processor SN. So the other shoe has yet to drop. And the other processor guys may not know what they will do yet.

Terry Ritter ritter@io.com http://www.io.com/~ritter/
Crypto Glossary http://www.io.com/~ritter/GLOSSARY.HTM

Subject: Re: Pentium III...
Date: Mon, 1 Feb 1999 11:09:00 GMT
From: sjmz@hplb.hpl.hp.com (Stefek Zaba)
Message-ID: <F6H2z0.5p@hplb.hpl.hp.com>
References: <36aaaad8.5611074@news.io.com>
Newsgroups: sci.crypt
Lines: 22

In sci.crypt, Terry Ritter (ritter@io.com) wrote:

> Producing cryptographic random values on a web server does not sound
> like a great idea to me.

A "great" idea it may not be, but - at a plausible guess - well over 90% of "cryptographic random numbers" in fielded use today are being generated on web servers and clients. I make this guesstimate on the grounds that the most widely-deployed crypto protocol on the Net is SSL/TLS, with HTTP as the dominant traffic carried by SSL. The session keys are generated by the client alone (SSLv2), or co-operatively by client and server (SSLv3/TLS). Depending on the platform, the degree of interaction with less predictable external events may be lesser or greater - /dev/random on Linux boxes, crufty combinations of (high-resolution) hard disk seek times and activities of other processes on Windows implementations. It's very rare for fielded e-commerce sites to run RNG hardware at the server end - and even if you do, there's no way for the server to influence the session key under SSLv2 :-(

Having an on-chip source of unpredictability to stir into the mix should be a help; it won't be if it's carelessly implemented or carelessly used, though.

Stefek

Subject: Re: Pentium III...
Date: Fri, 22 Jan 1999 12:17:00 -0600
From: jgfunj@vgrknf.arg (wtshaw)
Message-ID: <jgfunj-2201991217000001@dial-244-161.itexas.net>
References: <36a78709.23154884@nntp.ix.netcom.com>
Newsgroups: sci.crypt
Lines: 20

In article <36a78709.23154884@nntp.ix.netcom.com>, rcktexas@ix.netcom.com wrote:
>
> You would think the industry learned its lesson from the old days of
> closed architectures, key-based S/W, dongle keys, etc. I can just see
> Ziff Davis refusing to test anything that is Pentium III based.
>
Sometimes one must think in an obtuse way to see what one should be looking out for. If some vital part of a computer, say the bios in a new package that one just "had to have" also contained the internal battery to sustain it, a serial number could be installed at the same time the battery was purchased. I suppose one could simply add a switch of some sort to go between several such chips. Then, buried deep in the system, a check might be added to verify that a registered bios was installed....

Embedded id's are just another step in an endless series of trumphs and overtrumphs to put security on yours or someone else's terms.
--
A much to common philosophy:
It's no fun to have power....unless you can abuse it.

Subject: Re: Pentium III...
Date: Thu, 21 Jan 1999 23:03:12 +0000
From: Anthony Naggs <amn@ubik.demon.co.uk>
Message-ID: <JxTxdyAwI7p2Ewas@ubik.demon.co.uk>
References: <787vlr$3b4$1@nnrp1.dejanews.com>
Newsgroups: sci.crypt
Lines: 15

After much consideration ? decided to share these wise words:
>
>> Intel has announced that the Pentium III will have a built in hardware
>> random number generator, and individual serial number on each chip.
>
>The primary use for a processor serial number seems like it would be to
>enforce software licenses.

IMO The primary use for a unique processor serial number is for easy identification of stolen processors, i.e. through software without dismantling suspect PCs & removing heatsink assemblies.

--
BAD COMPUTER! That's my registry file you've trashed.

Subject: Re: Pentium III...
Date: Sun, 24 Jan 1999 18:27:18 GMT
From: William Hugh Murray <whmurray@sprynet.com>
Message-ID: <36AB6605.DBB48168@sprynet.com>
References: <36A676E3.E8D27B14@egg.chips.and.spam.com>
Newsgroups: sci.crypt
Lines: 47

This is a multi-part message in MIME format.
----------57A0885E6C73978430B690B7

Anthony Naggs wrote:
>
> After much consideration ? decided to share these wise words:
> >
> >> Intel has announced that the Pentium III will have a built in hardware
> >> random number generator, and individual serial number on each chip.

> >
> >The primary use for a processor serial number seems like it would be to
> >enforce software licenses.
>
> IMO The primary use for a unique processor serial number is for easy
> identification of stolen processors, i.e. through software without
> dismantling suspect PCs & removing heatsink assemblies.

How about as a quid quo pro for an export license for the RNG?
----------57A0885E6C73978430B690B7
 name="whmurray.vcf"
 filename="whmurray.vcf"

begin:vcard
n:Murray;William Hugh
tel;fax:800-690-7952
tel;home:203-966-4769
tel;work:203-761-3088
org:Deloitte Touche
adr:;;24 East Avenue, Suite 1362;New Canaan;Connecticut;06840;
version:2.1
email;internet:whmurray@sprynet.com
title:Executive Consultant, Information Security
fn:William Hugh Murray
end:vcard

----------57A0885E6C73978430B690B7--

Subject: Re: Pentium III...
Date: Tue, 26 Jan 1999 11:36:51 GMT
From: leadacid.remove-this@remove-this.hotmail.com (Myself)
Message-ID: <36ad2ff2.317075143@news.123.net>
References: <787vlr$3b4$1@nnrp1.dejanews.com>
Newsgroups: sci.crypt
Lines: 10

On Thu, 21 Jan 1999 19:37:32 GMT, thermal and electromagnetic action
caused janb@pmatrix.com's brain to produce the following pseudorandom
thought:
>The OS is licensed to a specific machine, which might be interpreted as a
>specific processor.

And in reply, my brain hashed together this little bit of thoughtsum:
What do we do about multiprocessor machines?

-Myself-

Subject: Re: Pentium III...
Date: Tue, 26 Jan 1999 16:05:24 +0000
From: Marty Levy <rwww60@email.sps.mot.com>
Message-ID: <36ADE7C3.1FBF146C@email.sps.mot.com>
References: <36ad2ff2.317075143@news.123.net>
Newsgroups: sci.crypt
Lines: 7

Does anyone know the mechanism Intel plans to use to put the infamous serial
numbers on Pentium III chips? I wasn't aware that Pentiums had any
non-volitaile memory (other than ROM) on board. The only practical systems I
can think of is to use a fuse or laser repair type scheme.

Subject: Re: Pentium III...
Date: Wed, 27 Jan 1999 07:09:41 -1000
From: handWave <shaken@stirred.com>
Message-ID: <36AF4855.60ED@stirred.com>
References: <36ADE7C3.1FBF146C@email.sps.mot.com>
Newsgroups: sci.crypt
Lines: 22

Marty Levy wrote:
>
> Does anyone know the mechanism Intel plans to use to put the infamous serial
> numbers on Pentium III chips? I wasn't aware that Pentiums had any
> non-volitaile memory (other than ROM) on board. The only practical systems I
> can think of is to use a fuse or laser repair type scheme.

When I worked at Intel I showed them how to make EPROM cells using the
ordinary microprocessor wafer fabrication process. In 1986 I drew the
"single poly EPROM cell" on the CAD system, had it processed on a test
wafer, tested it, and it worked. I told the marketing department about
it. I wrote it up in my patent notebook. I told them to use it as a
serial number for the 80386, for key storage and for fabrication lot
tracking for process analysis.

The first generation of EPROM cells during the 1970's also used a single
polycrystaline silicon layer. I have not seen the pentoid three, but I
expect that it uses this memory cell. It is better than a fuse because it

does not explode and crater the top oxides which protect the chip from chemical contamination. Ask Larry Palley at Intel. Or Kurt Robinson at Intel. They know I did these things, and they should send me a royalty check. They know my name, even if sci.crypt does not.

Subject: Re: Pentium III...
Date: Thu, 28 Jan 1999 01:54:58 +0100
From: fungus <spam@egg.chips.and.spam.com>
Message-ID: <36AFB562.D3E77F17@egg.chips.and.spam.com>
References: <36AF4855.60ED@stirred.com>
Newsgroups: sci.crypt
Lines: 20

handWave wrote:
>
> In 1986 I drew the
> "single poly EPROM cell" on the CAD system, had it processed on a test
> wafer, tested it, and it worked. I told the marketing department about
> it. I wrote it up in my patent notebook. I told them to use it as a
> serial number for the 80386, for key storage and for fabrication lot
> tracking for process analysis.
>

...so you're personally to blame for all this!

--
<___/>
/ O O \
_____/ FTB.

Subject: Re: Pentium III...
Date: 27 Jan 1999 09:58:11 -0500
From: Bruce Barnett <see.my.address.below@domain.com>
Message-ID: <yek3e4wsqyk.fsf@grymoire.crd.ge.com>
References: <787vlr$3b4$1@nnrp1.dejanews.com>
Newsgroups: sci.crypt
Lines: 11

janb@pmatrix.com writes:

> I can think of cases where having a machine serial number would be somewhat
> handy. I'm not sure I would build it into the processor though.

And what do you do with a 2,4,8 or 64-CPU server?
It's the big servers that need it.
Dynamic roll-over should be interesting...

--
Bruce <barnett at crd. ge. com> (speaking as myself, and not a GE employee)

Subject: Re: Pentium III...
Date: Thu, 28 Jan 1999 22:45:03 GMT
From: janb@pmatrix.com
Message-ID: <78qp9d$2v3$1@nnrp1.dejanews.com>
References: <yek3e4wsqyk.fsf@grymoire.crd.ge.com>
Newsgroups: sci.crypt
Lines: 21

In article <yek3e4wsqyk.fsf@grymoire.crd.ge.com>,
  Bruce Barnett <see.my.address.below@domain.com> wrote:
> janb@pmatrix.com writes:
>
> > I can think of cases where having a machine serial number would be somewhat
> > handy. I'm not sure I would build it into the processor though.
>
> And what do you do with a 2,4,8 or 64-CPU server?
> It's the big servers that need it.
> Dynamic roll-over should be interesting...

I don't see any difference between 1 or 64 processors, big servers or handheld devices. Pretty much every applications I can think of (except mabey reducing CPU theft) would want a unique identity (serial number) for each system. A system might even be a cluster. My preference would be to have that system identity survive cpu and hardware upgrades/changes.

- Jan

----------== Posted via Deja News, The Discussion Network ==----------
http://www.dejanews.com/ Search, Read, Discuss, or Start Your Own

---

Subject: Re: Pentium III...
Date: 03 Feb 1999 10:53:58 -0500
From: Bruce Barnett <see.my.address.below@domain.com>
Message-ID: <yekd83rqy95.fsf@grymoire.crd.ge.com>

References: <78qp9d$2v3$1@nnrp1.dejanews.com>
Newsgroups: sci.crypt
Lines: 32

janb@pmatrix.com writes:

> I don't see any difference between 1 or 64 processors, big servers or handheld
> devices. Pretty much every applications I can think of (except mabey reducing
> CPU theft) would want a unique identity (serial number) for each system. A
> system might even be a cluster. My preference would be to have that system
> identity survive cpu and hardware upgrades/changes.

Exactly. But putting the ID on the chip makes this difficult/
impossible.

I've talked to field server people that get upset when they have to
replace a disk drive, and the security ID changes, and breaks the
system. Same will be true in a fortified server. If the one chip that
has the master ID fails, and is swapped, the entire system becomes
invalid. You have to shut down the server and re-establish identities
between every system it has an association with.

So - picture a server with 100,000 clients. The clients trust the
server. The server fails, and needs a new CPU. The 100,000 clients no
longer trust the server.

Where is the documents that explain the algorithm in detail?
So far it seems to be vaporware. Security documents must be reviewed
by peers. Until they do, they are useless. It's even useless to
speculate about them.

--
Bruce <barnett at crd. ge. com> (speaking as myself, and not a GE employee)

Subject: Re: Pentium III...
Date: Fri, 22 Jan 1999 14:21:15 +0000
From: Guy Dawson <guy@crossflight.co.uk>
Message-ID: <36A8895B.752DCF1C@crossflight.co.uk>
References: <36A676E3.E8D27B14@egg.chips.and.spam.com>
Newsgroups: sci.crypt
Lines: 22

fungus wrote:
>
> Intel has announced that the Pentium III will have a built in hardware
> random number generator, and individual serial number on each chip.

This makes it much easier (well, possible) to determine if a chip
is one of a batch of stolen chips.

There have been quite a number of raid on truck carrying Intel CPUs.
They're currently easy to sell on the black market as they are
commodity items that can't be traced.

The rated CPU speed can also be recorded against the serial number
and this used to determine if a supplier is re-rating CPUs. That is,
taking a 333Mhz Celeron and passing it off as a 400MHz one.

Guy
-- ---
Guy Dawson I.T. Manager Crossflight Ltd
guy@crossflight.co.uk 0973 797819 01753 776104

Subject: Re: Pentium III...
Date: Fri, 22 Jan 1999 16:32:53 GMT
From: rcktexas@ix.netcom.com (R. Knauer)
Message-ID: <36a8a535.22005892@nntp.ix.netcom.com>
References: <36A8895B.752DCF1C@crossflight.co.uk>
Newsgroups: sci.crypt
Lines: 24

On Fri, 22 Jan 1999 14:21:15 +0000, Guy Dawson <guy@crossflight.co.uk>
wrote:

>This makes it much easier (well, possible) to determine if a chip
>is one of a batch of stolen chips.

>There have been quite a number of raid on truck carrying Intel CPUs.
>They're currently easy to sell on the black market as they are
>commodity items that can't be traced.

>The rated CPU speed can also be recorded against the serial number
>and this used to determine if a supplier is re-rating CPUs. That is,
>taking a 333Mhz Celeron and passing it off as a 400MHz one.

I guess the next thing is to keep a database on your DNA so you can be
followed everywhere you go. <jeez>

Bob Knauer

"It is not the function of our government to keep the citizen from
falling into error; it is the function of the citizen to keep the
government from falling into error."
--Justice Robert H. Jackson

Subject: Re: Pentium III...
Date: Sat, 23 Jan 1999 19:33:07 -0700
From: Robert Yoder <ryoder@tci.com>
Message-ID: <36AA8663.5160B4AE@tci.com>
References: <36A8895B.752DCF1C@crossflight.co.uk>
Newsgroups: sci.crypt
Lines: 57

Guy Dawson wrote:
>
> fungus wrote:
> >
> > Intel has announced that the Pentium III will have a built in hardware
> > random number generator, and individual serial number on each chip.
>
> This makes it much easier (well, possible) to determine if a chip
> is one of a batch of stolen chips.
>
> There have been quite a number of raid on truck carrying Intel CPUs.
> They're currently easy to sell on the black market as they are
> commodity items that can't be traced.
>
> The rated CPU speed can also be recorded against the serial number
> and this used to determine if a supplier is re-rating CPUs. That is,
> taking a 333Mhz Celeron and passing it off as a 400MHz one.

According to:

http://www4.tomshardware.com/releases/99q1/990121/cpu-news-01.html

"The new identification number is not targeted against processor
remarking
and Intel is not planning to provide a list where each identification
numbers
refers to the proper CPU speed. This number is not meant to fight
overclocking,
it's only meant to improve network security."

The official Intel blurb is toward the end of this lengthy example of
handwaving:

http://www.intel.com/pressroom/archive/speeches/pg012099.htm

Do yourself a favor and search the page for the word "serial" and
begin reading from there to the end. Being forced to read the
whole thing is a violation of the Geneva Convention.

Apparently, Intel seriously wants us to believe that having a unique
number embedded in your CPU is a giant leap in network security for
user authentication. Well since the only thing that goes out on the
wire, is what the OS puts on that wire, and the OS is composed of
SOFTWARE, not _HARDWARE_, I submit that the whole thing has less
validity than cold fusion.

I suspect that the CPU ID was driven by large commerical software
vendors who wanted a way to node-lock their licenses, and Intel is
trying cover the whole thing with a sugar-coating and convince the
consumers that this is good for us.

Robert Yoder
--
ryoder@tci.com
"Unix: The Solution to the W2K Problem."

Subject: Re: Pentium III...
Date: 24 Jan 1999 17:21:56 -0800
From: mskala@ansuz.sooke.bc.ca. (Matthew Skala)
Message-ID: <78ggvk$23i$1@ruby.ansuz.sooke.bc.ca>
References: <36AA8663.5160B4AE@tci.com>
Newsgroups: sci.crypt
Lines: 22

In article <36AA8663.5160B4AE@tci.com>, Robert Yoder <ryoder@tci.com> wrote:
>I suspect that the CPU ID was driven by large commerical software
>vendors who wanted a way to node-lock their licenses, and Intel is
>trying cover the whole thing with a sugar-coating and convince the
>consumers that this is good for us.

It's good for me. I want to build large parallel processors; if each CPU
has its own ID number, that may make it easier for then to talk to each
other without interferance. I don't really need it, I can get the same

functionality in other ways, but it has a small nonzero dollar value.
I'm not afraid of it being used to identify me - ethernet cards, PC cases, and hard drives all have serial numbers on them already anyway. The CPU ID is less threatening because nobody's going to be able to read my CPU ID number without first being able to execute programs on my machine. If someone is so stupid as to use my CPU ID (or more properly, what my machine *claims* is its CPU ID) for network security, then I can have some real fun.
--
The third girl had an upside-down penguin on her stomach, so the doctor told her, "I'll examine you for free, if you and your boyfriend will debug my Web server."

Matthew Skala
Ansuz BBS
(250) 472-3169
http://www.islandnet.com/~mskala/

Subject: Re: Pentium III...
Date: Sun, 24 Jan 1999 19:43:42 -0700
From: Robert Yoder <ryoder@tci.com>
Message-ID: <36ABDA5E.7D961BCC@tci.com>
References: <78ggvk$23i$1@ruby.ansuz.sooke.bc.ca>
Newsgroups: sci.crypt
Lines: 70

Matthew Skala wrote:
>
> In article <36AA8663.5160B4AE@tci.com>, Robert Yoder <ryoder@tci.com> wrote:
> >I suspect that the CPU ID was driven by large commerical software
> >vendors who wanted a way to node-lock their licenses, and Intel is
> >trying cover the whole thing with a sugar-coating and convince the
> >consumers that this is good for us.
>
> It's good for me. I want to build large parallel processors; if each CPU
> has its own ID number, that may make it easier for then to talk to each
> other without interferance. I don't really need it, I can get the same
> functionality in other ways, but it has a small nonzero dollar value.

Now I don't have any experience with multi-CPU Intel machines,
but I _DO_ have experience with multi-CPU SPARC machines, and
in that environment, every CPU has it's own ID based on _WHERE_
it is plugged in, and not on a CPU-specific hard-coded number.
e.g.

```
$ psrinfo -v
Status of processor 0 as of: 01/24/99 19:27:11
  Processor has been on-line since 01/15/99 16:57:08.
  The sparc processor operates at 248 MHz,
        and has a sparc floating point processor.
Status of processor 1 as of: 01/24/99 19:27:11
  Processor has been on-line since 01/15/99 16:57:12.
  The sparc processor operates at 248 MHz,
        and has a sparc floating point processor.
Status of processor 4 as of: 01/24/99 19:27:11
  Processor has been on-line since 01/15/99 16:57:12.
  The sparc processor operates at 248 MHz,
        and has a sparc floating point processor.
Status of processor 5 as of: 01/24/99 19:27:11
  Processor has been on-line since 01/15/99 16:57:12.
  The sparc processor operates at 248 MHz,
        and has a sparc floating point processor.
$
```

In this machine, (E6000), board slots 0 and 2 contain CPU/memory boards,
and each board hold 2 CPU's. Surely an Intel machine can do something
equivalent w/o hard-coding a number into the CPU.

> I'm not afraid of it being used to identify me - ethernet cards, PC cases,
> and hard drives all have serial numbers on them already anyway. The CPU ID
> is less threatening because nobody's going to be able to read my CPU ID
> number without first being able to execute programs on my machine. If
> someone is so stupid as to use my CPU ID (or more properly, what my
> machine *claims* is its CPU ID) for network security, then I can have some
> real fun.

But in the case of MAC-addresses, we haven't had a huge corporation
trying to tell us, (with a straight face), that the MAC address can
be used for user authentification. Sun has had a "hostid" embedded
in a MB chip for ages which is used for s/w licensing, and HP has
had a similar feature for years, but neither of them have ever tried
to tell us this could be used for user authentication. (And if you
look around the net you can find a s/w program that will circumvent
the Sun hostid.)

I don't have any _FEAR_ of the CPU-ID "feature"; I just resent having
my intelligence insulted by Intel lying to tell us about what it can
be used for. And since I don't use third-rate proprietary OS's, I
have no worry that my CPU-ID could ever be transmitted w/o my
knowing.

ry
--
ryoder@tci.com
"Unix: The Solution to the W2K Problem."

```
Subject: Re: Pentium III...
Date: Mon, 25 Jan 1999 08:55:54 -0700
From: Robert Yoder &LT;ryoder@tci.com>
Message-ID: &LT;36AC940A.2DF22C1A@tci.com>
References: &LT;36ABDA5E.7D961BCC@tci.com>
Newsgroups: sci.crypt
Lines: 15

Check this out:

http://www.news.com/News/Item/0,4,31354,00.html?st.cn.fd.tkr.ne

Two different organizations are organizing a boycott of Intel
because the CPU ID is "an invasion of privacy".

If Intel hadn't lied to us about the purpose of the
CPU ID, there wouldn't even _BE_ a privacy issue!


ry
--
ryoder@tci.com
"Unix:  The solution to the W2K Problem."
```

Terry Ritter, *his current address*, *and his top page*.

Last updated: 1999-02-21

Random Numbers from a Sound Card

A *Ciphers By Ritter* Page

Everybody has a sound card, so we all have a physically-random noise generator -- a source of absolute randomness -- right? A discussion starting with sound cards recording noise, and ending with theories of randomness.

Contents

- 1999-01-25 Mok-Kong Shen: "There are tests for statistical quality, e.g. Maurer's universal statistical test. I am ignorant of tests for crypto quality."
- 1999-01-25 Paul Rubin: "We just test the total amount of energy in the audio to make sure the mic isn't dead. We expect that the raw audio will have lots of correlation, so we run it through a hash function or block cipher...."
- 1999-01-26 Nathan Kennedy: "I tune a cheap AM radio to a loud static channel, and wire that into the mic port."
- 1999-01-26 Cuykens Anthony: "I do remember of a way a teacher told me to generate a 'true' random generator." "You select some measurable information about your noise source.... Then you sample you source at fixed interval.... For each coosen information, if it is higher than the same info at the last sample, the output is one, otherwize the result is zero."
- 1999-01-26 Jon Haugsand: "Better is to make two samples not too close in time and output a one if the first is higher, and output a zero if the second is higher."
- 1999-01-26 randombit@my-dejanews.com: "Your teacher was just trying to communicate that you need a physical signal to start with...."
- 1999-01-27 R. Knauer: "You got a computer that can roll dice...?"
- 1999-01-26 Nathan Kennedy: "The best approach is not to waste any entropy, and just feed the raw data to a hungry hash function...."
- 1999-01-26 David Ross: "I create one rotor at a time...."
- 1999-01-26 Frank Gifford: "You might want to look into generating a simple rotor and use the random numbers to swap entries in the rotor."
- 1999-01-26 Paul Rubin: "Don't even think of using raw soundblaster output as actual random data rather than just as an entropy source.
- 1999-01-26 Kazak, Boris: "...HH is a Housing (just a glass or plastic bottle), OO are... 100-200 peas or beans, MM is a Microphone. Now if we start rotating the Housing...."
- 1999-01-27 R. Knauer: "Any recommendations on which hash function or something similar?"
- 1999-01-27 randombit@my-dejanews.com: "In order to know how many bits you have to distill with MD5, or any other hash, you need to measure your entropy/raw bit."
- 1999-01-27 randombit@my-dejanews.com: "FM has better hiss. I leave it to RF folks to explain why."
- 1999-01-27 David Ross: "In my case, the FM limiter & detector puts out a waveshape which is more linear in the voltage range where I'm digitizing."
- 1999-01-26 R. Knauer: "It is a fundamental precept of crytpography that randomness is a property of how a number is generated, not a property of a number itself."
- 1999-01-26 Mok-Kong Shen: "...OTP presuppose (absolutely) true randomness and there is no way of determining that in practice."
- 1999-01-26 R. Knauer: "For all practical purposes the OTP is proveably secure."
- 1999-01-27 Mok-Kong Shen: "I am not against having something ideal and perfect as a standard for approximation.... But to say there IS... something perfect can be misleading."
- 1999-01-27 R. Knauer: "Does a Perfect Circle EXIST?"
- 1999-01-27 Mok-Kong Shen: "If the word 'IS' is employed in a context without the connotation of 'EXISTS' then it is NOT misleading, otherwise it IS misleading."
- 1999-01-27 R. Knauer: "It all depends on what the meaning of the word 'is' is."
- 1999-01-27 Mok-Kong Shen: "A word can have a multitude of meanings. I was prudent enough to put the parentheses above to make sure that there could be no misunderstanding."
- 1999-01-27 R. Knauer: "You must be a mathematician."
- 1999-01-27 Medical Electronics Lab: "We can certainly build a TRNG which is perfect in any measureable sense."
- 1999-01-27 Tony T. Warnock: "In limit (infinitely long sequences) both the complexity based and the frequency (statistical) based definitions of random are equivalent...."
- 1999-01-27 R. Knauer: "Those things don't measure the crypto-grade randomness of finite numbers at all."
- 1999-01-28 Medical Electronics Lab: "So you need an infinite sequence of bits to prove that something is crypto-grade random, yes?"
- 1999-01-28 R. Knauer: "The only way you can prove the crypto-grade randomness of a finite number is to consider the method of generation."
- 1999-01-29 Mok-Kong Shen: "Ah! Finally one knows exactly what the term 'crypto-grade random numbers' you employ means: These are DEFINED to be the output from a hardware generator."
- 1999-01-29 R. Knauer: "A TRNG is not a True Random Number Generator just because it is a hardware device."
- 1999-01-29 Patrick Juola: "Not all hardware generators are TRNG."
- 1999-01-29 Tony T. Warnock: "Complexity of computation and statistical properties are only equivalent in the limit of infnitely many infinitely long sequences."
- 1999-01-30 R. Knauer: "But I point out that computational complexity has nothing fundamental to do with crypto-grade randomness, nor QM."
- 1999-01-30 R. Knauer: "How come you state that Champernowne's number has an 'excess of ones over zeros'?"
- 1999-01-30 Trevor Jackson, III: "Because all the leading zeros are suppressed."
- 1999-01-31 R. Knauer: "Can you elaborate with an example."
- 1999-01-26 Patrick Juola: "...you'll probably not build a 'perfect' OTP in practice...."
- 1999-01-26 Paul L. Allen: "I'd be *extremely* worried about a sound card (particularly with little in the way of input) picking up mains power hum and radiated noise from signal lines in the computer." "Noise diodes strike me as being safer."
- 1999-01-26 randombit@my-dejanews.com: "I experimented with parity-of-N bits, and used Maurer's Universal statistical test for RNGs to measure the entropy. When you distill enough, the entropy reaches its expected value."
- 1999-01-26 David Ross: "If it has a 1 Volt peak amplitude and if I digitize at a constant rate, won't 1/2 of my samples yield a value of either >+.707 Volt or < -.707 Volt?"
- 1999-01-27 randombit@my-dejanews.com: "The threshold of your detector, which decides whether a voltage is to be called a 0 or a 1...."
- 1999-01-28 randombit@my-dejanews.com: "Eve does not know the local, instantaneious electromagnetic conditions around my receiver, nor does she know what my local electronics are doing."
- 1999-01-28 Mok-Kong Shen: "I believe that under certain real circumstances obtaining bit sequences from software can be justified."
- 1999-01-28 patix: "Haw should we test hardawre random generator to "be shoure" that it is realy some haw random ?"
- 1999-01-29 Patrick Juola: "No, but Mike does. He's fully capable of broadcasting (known) noise of some sort near your site."
- 1999-01-26 Paul L. Allen: "I can never find a URL for that when I need it."

```
Subject: Re: Random numbers from a sound card?
Date: Mon, 25 Jan 1999 20:11:33 +0100
From: Mok-Kong Shen &LT;mok-kong.shen@stud.uni-muenchen.de>
Message-ID: &LT;36ACC1E5.90C4C2BC@stud.uni-muenchen.de>
References: &LT;36acb8b1.5374650@news.willapabay.org>
Newsgroups: sci.crypt
Lines: 15
```

David Ross wrote:
>

> How would you test the 'quality' of the generated random number
> stream?

There are tests for statistical quality, e.g. Maurer's universal
statistical test. I am ignorant of tests for crypto quality.
I guess the issue of cryptological strength is inherently fuzzy
and not entirely separable from subjectivity and concepts like
confidence intervals, i.e. no security can be claimed on an absolute
scale in practice. But experts might refute my un-knowledgeable
assertions.

M. K. Shen

Subject: Re: Random numbers from a sound card?
Date: Mon, 25 Jan 1999 21:21:09 GMT
From: phr@netcom.com (Paul Rubin)
Message-ID: <phrF64wn9.J9M@netcom.com>
References: <36acb8b1.5374650@news.willapabay.org>
Newsgroups: sci.crypt
Lines: 20

In article <36acb8b1.5374650@news.willapabay.org>,
David Ross <ross@hypertools.com> wrote:
> Has anyone had success using a sound card (like a Sound Blaster) to
>generate streams of random numbers?

Yes, see http://www.lila.com/nautilus/index.html and download the
source from one of the sites mentioned there.

> What sort of audio source would you suspect would be the best to use
>in generating random numbers?

We ask the user to blow into the microphone to make noise, IIRC.

> How would you test the 'quality' of the generated random number
>stream?

We just test the total amount of energy in the audio to make sure
the mic isn't dead. We expect that the raw audio will have lots
of correlation, so we run it through a hash function or block cipher;
I don't remember the details.

Subject: Re: Random numbers from a sound card?
Date: Tue, 26 Jan 1999 07:09:28 +0800
From: Nathan Kennedy <blaaf@hempseed.com>
Message-ID: <36ACF9A8.CF6CF898@hempseed.com>
References: <36acb8b1.5374650@news.willapabay.org>
Newsgroups: sci.crypt
Lines: 32

David Ross wrote:
>
> Has anyone had success using a sound card (like a Sound Blaster) to
> generate streams of random numbers?

Sure. My favorite (T)RNG method.

>
> What sort of audio source would you suspect would be the best to use
> in generating random numbers?

I tune a cheap AM radio to a loud static channel, and wire that into the
mic port.

> How would you test the 'quality' of the generated random number
> stream?

Of course, you can't test the 'quality' by looking at the random numbers
generated. You need to estimate the entropy of your source, and of course
it's always going to be an estimate, you can almost never prove it.

What I did, was compress it, multiply my hash size by the compression ratio
by a fudge factor of 10. Then I would hash that much data, and assumed
that the result was very close to 100% entropy. This is rather paranoid
and slow though. If you don't need 100% entropy just go ahead and
continually sample /dev/audio for data and use it as entropy for a PRNG,
and sample the PRNG as often as you like. The quality should still be
excellent... As long as you've got >128 bits of entropy total and the PRNG
does its job, the result should be quite secure as long as nothing gets
compromised.

Nate

Subject: Re: Random numbers from a sound card?
Date: Tue, 26 Jan 1999 09:41:42 +0100
From: Cuykens Anthony <cuykens.a@protonworld.com>

Message-ID: <36AD7FC6.674ABAEC@protonworld.com>
References: <36ACF9A8.CF6CF898@hempseed.com>
Newsgroups: sci.crypt
Lines: 53

Hi,

I do remember of a way a teacher told me to generate a "true" random generator. You select some measurable information about your noise source (in your case, lets says the frequence, the loudness, ...). Then you sample you source at fixed interval and you check all your informations. For each coosen information, if it is higher than the same info at the last sample, the output is one, otherwize the result is zero. At each sample, this method will give you one bit per criterion.

This is just an idea, what does guru think of it?

Nathan Kennedy wrote:

> David Ross wrote:
> >
> > Has anyone had success using a sound card (like a Sound Blaster) to
> > generate streams of random numbers?
>
> Sure. My favorite (T)RNG method.
>
> >
> > What sort of audio source would you suspect would be the best to use
> > in generating random numbers?
>
> I tune a cheap AM radio to a loud static channel, and wire that into the
> mic port.
>
> > How would you test the 'quality' of the generated random number
> > stream?
>
> Of course, you can't test the 'quality' by looking at the random numbers
> generated. You need to estimate the entropy of your source, and of course
> it's always going to be an estimate, you can almost never prove it.
>
> What I did, was compress it, multiply my hash size by the compression ratio
> by a fudge factor of 10. Then I would hash that much data, and assumed
> that the result was very close to 100% entropy. This is rather paranoid
> and slow though. If you don't need 100% entropy just go ahead and
> continually sample /dev/audio for data and use it as entropy for a PRNG,
> and sample the PRNG as often as you like. The quality should still be
> excellent... As long as you've got >128 bits of entropy total and the PRNG
> does its job, the result should be quite secure as long as nothing gets
> compromised.
>
> Nate

--
Anthony Cuykens

Subject: Re: Random numbers from a sound card?
Date: 26 Jan 1999 10:18:04 +0100
From: Jon Haugsand <haugsand@procyon.nr.no>
Message-ID: <yzo90eqv1df.fsf@procyon.nr.no>
References: <36AD7FC6.674ABAEC@protonworld.com>
Newsgroups: sci.crypt
Lines: 26

* Cuykens Anthony
| I do remember of a way a teacher told me to generate a "true" random
| generator. You select some measurable information about your noise source (in
| your case, lets says the frequence, the loudness, ...). Then you sample you
| source at fixed interval and you check all your informations. For each coosen
| information, if it is higher than the same info at the last sample, the output
| is one, otherwize the result is zero. At each sample, this method will give you
| one bit per criterion.

I am not sure that this will be random enough. I would guess that as the number of concectutive ones increases, the probability to get a zero the next time also increases. Better is to make two samples not too close in time and output a one if the first is higher, and output a zero if the second is higher.

Better still is to measure some quantity twice (e.g. sound level) and use the least significant bit and output a one if you measure 10, output a zero if you get 01. If you get 11 or 00, discard those.

--
Jon Haugsand
Norwegian Computing Center, <http://www.nr.no/engelsk/>
<mailto:haugsand@nr.no> Pho: +47 22852608 / +47 22852500,
Fax: +47 22697660, Pb 114 Blindern, N-0314 OSLO, Norway

Subject: Re: Random numbers from a sound card?
Date: Tue, 26 Jan 1999 20:47:01 GMT
From: randombit@my-dejanews.com
Message-ID: <78l9k2dsh1@nnrp1.dejanews.com>
References: <yzo90eqv1df.fsf@procyon.nr.no>
Newsgroups: sci.crypt
Lines: 14

In article <yzo90eqv1df.fsf@procyon.nr.no>,
  Jon Haugsand <haugsand@procyon.nr.no> wrote:
> * Cuykens Anthony
> | I do remember of a way a teacher told me to generate a "true" random
> | generator.

Your teacher was just trying to communicate that you need a physical
signal to start with, as deterministic algorithms can't produce
unpredictability (aka randomness).

Computers that can roll dice are not Turing machines.

----------== Posted via Deja News, The Discussion Network ==----------
http://www.dejanews.com/ Search, Read, Discuss, or Start Your Own

---

Subject: Re: Random numbers from a sound card?
Date: Wed, 27 Jan 1999 01:33:31 GMT
From: rcktexas@ix.netcom.com (R. Knauer)
Message-ID: <36ae6c8a.51870215@nntp.ix.netcom.com>
References: <78l9k2dsh1@nnrp1.dejanews.com>
Newsgroups: sci.crypt
Lines: 14

On Tue, 26 Jan 1999 20:47:01 GMT, randombit@my-dejanews.com wrote:

>Computers that can roll dice are not Turing machines.

You got a computer that can roll dice - a completetly
non-deterministic machine that can compute algorithmically?

Bob Knauer

"An honest man can feel no pleasure in the exercise of power over
his fellow citizens."
--Thomas Jefferson

Subject: Re: Random numbers from a sound card?
Date: Tue, 26 Jan 1999 17:19:57 +0800
From: Nathan Kennedy <blaaf@hempseed.com>
Message-ID: <36AD88BD.3397761D@hempseed.com>
References: <36AD7FC6.674ABAEC@protonworld.com>
Newsgroups: sci.crypt
Lines: 30

Cuykens Anthony wrote:
>
>   Hi,
>
>   I do remember of a way a teacher told me to generate a "true" random
> generator. You select some measurable information about your noise source (in
> your case, lets says the frequence, the loudness, ...). Then you sample you
> source at fixed interval and you check all your informations. For each coosen
> information, if it is higher than the same info at the last sample, the output
> is one, otherwize the result is zero. At each sample, this method will give you
> one bit per criterion.
>
>   This is just an idea, what does guru think of it?
>

That's just one way of converting a raw sampled value into a bit stream...
It doesn't assure any randomness. It would probably be very predictable in
the short term, and likely biased as well.

Certainly, applying this to soundcard sampled data is little better than
the raw output of /dev/audio.

The best approach is not to waste any entropy, and just feed the raw data
to a hungry hash function, which will process it into an unbiased output.
The hash never has more entropy than what it is seeded with, however!

Bruce Schneier has an excellent paper on PRNGs on his site
(www.counterpane.com), which could serve as a good introduction.

Nate

Subject: Re: Random numbers from a sound card?
Date: Tue, 26 Jan 1999 21:26:38 GMT
From: ross@hypertools.com (David Ross)

Message-ID: <36ae3069.15236697@news.willapabay.org>
References: <78l32u$fr4@trebuchet.eng.us.uu.net>
<36ae089d.5049458@news.willapabay.org>
<36AD7FC6.674ABAEC@protonworld.com>
Newsgroups: sci.crypt
Lines: 35

On 26 Jan 1999 13:55:26 -0500, giff@eng.us.uu.net (Frank Gifford)
wrote:

>In article <36ae089d.5049458@news.willapabay.org>,
>David Ross <ross@hypertools.com> wrote:
>> Have tried something very similar to that. I am attempting to
>>create a rotortable of all 256 byte values placed in 'random' order,
>>but the (8 bit) SoundBlaster seems reluctant to produce a 0xC0 byte.
>>I infer this because in over 80% of the rotortables I create, 0xC0 is
>>the last table entry.
>
>How are you creating the tables? I would assume that since you are creating
>rotors, that each value appears in the rotor exactly once. Are you swapping
>values or some other method? Personally, I would suspect your creation
>routine is not doing what you want instead of bad numbers.

Giff -

I create one rotor at a time, waiting for each one of the 256
bytevalues to come in from the SoundBlaster before I go on to the next
rotor. A very simple piece of code, done in assembly language.

- several bytes commonly occurred toward the end of each rotor, but
0xC0 was by far the most popular as the last byte. (Incidentally, I'm
using an ES1688 sound chip set up to emulate a SoundBlaster.)

- the process of rotor creation took _much_ more time than I had
estimated.

- using a 'small' (50+) rotor encryption scheme to encrypt the
SoundBlaster bytes before sending them to the rotor sorting routine
sped up the process by about 20X.

David Ross ross@hypertools.com

Subject: Re: Random numbers from a sound card?
Date: 26 Jan 1999 17:30:58 -0500
From: giff@eng.us.uu.net (Frank Gifford)
Message-ID: <78lfn2$g6t@trebuchet.eng.us.uu.net>
References: <36ae3069.15236697@news.willapabay.org>
Newsgroups: sci.crypt
Lines: 44

In article <36ae3069.15236697@news.willapabay.org>,
David Ross <ross@hypertools.com> wrote:
> I create one rotor at a time, waiting for each one of the 256
>bytevalues to come in from the SoundBlaster before I go on to the next
>rotor. A very simple piece of code, done in assembly language.

Does this mean that (for a given rotor) you loop through rotor positions
and get a value from SB that has not been used yet, and then do the
remaining positions that way? So if SB gives you the same byte several
times in a row that you ignore the duplicates?

> - several bytes commonly occurred toward the end of each rotor, but
>0xC0 was by far the most popular as the last byte.

Assuming I understand your process, that means 0xC0 is very unlikely in
a byte stream. In that case, SB in your set up is probably a bad choice
for a random number generator. You may have to investigate your set up
a bit more. Not enough static for input?

> - the process of rotor creation took _much_ more time than I had
>estimated.

If you are doing it the way above, then yes indeed. When you get to the last
two values, you are waiting for either of them to be generated so you can
fill in the last piece of the rotor. You might want to look into generating
a simple rotor and use the random numbers to swap entries in the rotor. Then
you can simplify your code and generate a new rotor in a known amount of time.

> - using a 'small' (50+) rotor encryption scheme to encrypt the
>SoundBlaster bytes before sending them to the rotor sorting routine
>sped up the process by about 20X.

Does this mean you take the values from SB, pipe through your rotors, and
then use the results to create/modify a rotor? In that case, this may
be the source of weird results.

I would recommend checking your set up of SB to see whether the bytes it
generates directly is really 'random' and not biased.

-Giff

--
giff@uu.net Too busy for a .sig

Subject: Re: Random numbers from a sound card?
Date: Tue, 26 Jan 1999 22:20:11 GMT
From: phr@netcom.com (Paul Rubin)
Message-ID: <phrF66u1o.JBE@netcom.com>
References: <36ae089d.5049458@news.willapabay.org>
<36AD7FC6.674ABAEC@protonworld.com>
Newsgroups: sci.crypt
Lines: 24

In article <36ae089d.5049458@news.willapabay.org>,
David Ross <ross@hypertools.com> wrote:
>Nathan Kennedy wrote:
>
>> > What sort of audio source would you suspect would be the best to use
>> > in generating random numbers?
>>
>> I tune a cheap AM radio to a loud static channel, and wire that into the
>> mic port.
> Have tried something very similar to that. I am attempting to
>create a rotortable of all 256 byte values placed in 'random' order,
>but the (8 bit) SoundBlaster seems reluctant to produce a 0xC0 byte.
>I infer this because in over 80% of the rotortables I create, 0xC0 is
>the last table entry.
>
> I'd guess that the 'consumer-grade' A->D & D->A converters used
>in common sound cards are susceptible to all sorts of troubles like
>this, i.e. missing codes and/or monotonicity problems.

Don't even think of using raw soundblaster output as actual random
data rather than just as an entropy source. Even if the a/d converter
is terrific, it's still likely to pick up correlated noise from
various sources in the PC. Run the audio bits through a cryptographic
hash function or something similar before using it.

Subject: Re: Random numbers from a sound card?
Date: Tue, 26 Jan 1999 18:31:52 -0500
From: "Kazak, Boris" <bkazak@erols.com>
Message-ID: <36AE5068.7AD8@erols.com>
References: <phrF66u1o.JBE@netcom.com>
Newsgroups: sci.crypt
Lines: 56

Paul Rubin wrote:
>
> In article <36ae089d.5049458@news.willapabay.org>,
> David Ross <ross@hypertools.com> wrote:
> >Nathan Kennedy wrote:
> >
> >> > What sort of audio source would you suspect would be the best to use
> >> > in generating random numbers?
> >>
> >> I tune a cheap AM radio to a loud static channel, and wire that into the
> >> mic port.
> > Have tried something very similar to that. I am attempting to
> >create a rotortable of all 256 byte values placed in 'random' order,
> >but the (8 bit) SoundBlaster seems reluctant to produce a 0xC0 byte.
> >I infer this because in over 80% of the rotortables I create, 0xC0 is
> >the last table entry.
> >
> > I'd guess that the 'consumer-grade' A->D & D->A converters used
> >in common sound cards are susceptible to all sorts of troubles like
> >this, i.e. missing codes and/or monotonicity problems.
>
> Don't even think of using raw soundblaster output as actual random
> data rather than just as an entropy source. Even if the a/d converter
> is terrific, it's still likely to pick up correlated noise from
> various sources in the PC. Run the audio bits through a cryptographic
> hash function or something similar before using it.

Let's be practical...
  It is perfectly possible to use the sound card for random number
generation if we come up with a way to provide a random acoustic
input on its microphone connector.
  Consider such a simple system:

```
                HHHHHHHHHHHHHHHH
              HH               H   MMM
            HH                 H   MMMMM
            HH     OOOOOOOOOO  H   MMMMM
              HH  OOOOOOOOOOOO H   MMM
                HHHHHHHHHHHHHHHH
```

where HH is a Housing (just a glass or plastic bottle), OO are Objects
(a pseudo-scientific baptism for 100-200 peas or beans), MM is a
Microphone.
  Now if we start rotating the Housing around its horizontal axis,
the Objects will produce a loud Random Rattle, and the Microphone will
transmit this rattle to the sound card. My questions are:

How many Objects are needed and what must be the speed of rotation that will assure the True Randomness?
What estimates can be given for Degree of Correlation and for Period of Repetition, depending on the system parameters?

The System is not patented, it is hereby placed in the public domain.

Respectfully BNK

Subject: Re: Random numbers from a sound card?
Date: Wed, 27 Jan 1999 01:34:38 GMT
From: rcktexas@ix.netcom.com (R. Knauer)
Message-ID: <36ae6d00.51988195@nntp.ix.netcom.com>
References: <phrF66u1o.JBE@netcom.com>
Newsgroups: sci.crypt
Lines: 14

On Tue, 26 Jan 1999 22:20:11 GMT, phr@netcom.com (Paul Rubin) wrote:

>Run the audio bits through a cryptographic
>hash function or something similar before using it.

Any recommendations on which hash function or something similar?

Bob Knauer

"An honest man can feel no pleasure in the exercise of power over his fellow citizens."
--Thomas Jefferson

Subject: Re: Random numbers from a sound card?
Date: Wed, 27 Jan 1999 18:31:02 GMT
From: randombit@my-dejanews.com
Message-ID: <78nm16cvg1@nnrp1.dejanews.com>
References: <36ae6d00.51988195@nntp.ix.netcom.com>
Newsgroups: sci.crypt
Lines: 46

In article <36ae6d00.51988195@nntp.ix.netcom.com>,
  rcktexas@ix.netcom.com wrote:
> On Tue, 26 Jan 1999 22:20:11 GMT, phr@netcom.com (Paul Rubin) wrote:
>
> >Run the audio bits through a cryptographic
> >hash function or something similar before using it.
>
> Any recommendations on which hash function or something similar?
>
> Bob Knauer

You'll be told to use an established one, say MD5. Doesn't matter too much.

But what you *must* be careful with is this:

A sample output of MD5 will look random -that's its job :-)
It will appear to have full entropy.

In order to know how many bits you have to distill with MD5, or any other hash, you need to measure your entropy/raw bit.

Then you can convince skeptics that you are driving the hash with enough entropy.

If your hash function is *not* a crypto-strong one, then you can directly measure the quality of its output ---since its not crypto strong, by definition its output (when given very redundant input) will be crappy. Parity-of-N has this property. When N is large enough, for a given raw-entropy-rate, the parity output is indistinguishable from crypto-strong (ie, uniformly distributed) output. When N is insufficient, you can see it with an entropy measure.

"Many tame, conformist types felt the need to describe anti-social actions as 'sick'." -Ted K

----------== Posted via Deja News, The Discussion Network ==----------
http://www.dejanews.com/ Search, Read, Discuss, or Start Your Own

---

Subject: Re: Random numbers from a sound card?
Date: Wed, 27 Jan 1999 18:18:17 GMT

From: randombit@my-dejanews.com
Message-ID: <78nl8u$ca5$1@nnrp1.dejanews.com>
References: <36ae089d.5049458@news.willapabay.org>
<36AD7FC6.674ABAEC@protonworld.com>
Newsgroups: sci.crypt
Lines: 37

In article <36ae089d.5049458@news.willapabay.org>,
ross@hypertools.com (David Ross) wrote:
> Nathan Kennedy wrote:
>
> > > What sort of audio source would you suspect would be the best to use
> > > in generating random numbers?
> >
> > I tune a cheap AM radio to a loud static channel, and wire that into the
> > mic port.

FM has better hiss. I leave it to RF folks to explain why. Probably
because FM listens to wider chunks of the aether than AM. Or because
of the design of FM receivers amplifying component-noise more?

> Have tried something very similar to that. I am attempting to
> create a rotortable of all 256 byte values placed in 'random' order,
> but the (8 bit) SoundBlaster seems reluctant to produce a 0xC0 byte.
> I infer this because in over 80% of the rotortables I create, 0xC0 is
> the last table entry.
>
> I'd guess that the 'consumer-grade' A->D & D->A converters used
> in common sound cards are susceptible to all sorts of troubles like
> this, i.e. missing codes and/or monotonicity problems.
>
> David Ross ross@hypertools.com

Yes. You have to assume everything is imperfect. Your raw source
is biassed, your whole amplification/detection (ie, digitization) chain
has got holes in it (in either time or frequency domains), and
you're in a fun digital-switching environment for extra bonus
problems.

This is why measurement is so much better than handwaving.

----------== Posted via Deja News, The Discussion Network ==----------
http://www.dejanews.com/ Search, Read, Discuss, or Start Your Own

---

Subject: Re: Random numbers from a sound card?
Date: Wed, 27 Jan 1999 20:47:32 GMT
From: ross@hypertools.com (David Ross)
Message-ID: <36af78e0.12951524@news.willapabay.org>
References: <78nl8u$ca5$1@nnrp1.dejanews.com>
Newsgroups: sci.crypt
Lines: 22

On Wed, 27 Jan 1999 18:18:17 GMT, randombit@my-dejanews.com wrote:

>In article <36ae089d.5049458@news.willapabay.org
>ross@hypertools.com (David Ross) wrote:
> What sort of audio source would you suspect would be the best to use
>in generating random numbers?
>
>>> I tune a cheap AM radio to a loud static channel, and wire that into the
>>> mic port.
>
>>FM has better hiss. I leave it to RF folks to explain why. Probably
>>because FM listens to wider chunks of the aether than AM. Or because
>>of the design of FM receivers amplifying component-noise more?

  I see this 'better hiss' quality too, and suspect that it is due to
the FM detection scheme.

  In my case, the FM limiter & detector puts out a waveshape which is
more linear in the voltage range where I'm digitizing. This yields a
flatter distribution of A->D output bytes but is a bit slower.

 David Ross ross@hypertools.com

Subject: Re: Random numbers from a sound card?
Date: Tue, 26 Jan 1999 00:28:16 GMT
From: rcktexas@ix.netcom.com (R. Knauer)
Message-ID: <36ad09fc.46447347@nntp.ix.netcom.com>
References: <36ACC1E5.90C4C2BC@stud.uni-muenchen.de>
Newsgroups: sci.crypt
Lines: 35

On Mon, 25 Jan 1999 20:11:33 +0100, Mok-Kong Shen
<mok-kong.shen@stud.uni-muenchen.de> wrote:

>> How would you test the 'quality' of the generated random number
>> stream?

>There are tests for statistical quality, e.g. Maurer's universal
>statistical test. I am ignorant of tests for crypto quality.

That's because there aren't any.

It is a fundamental precept of crytpography that randomness is a property of how a number is generated, not a property of a number itself.

>I guess the issue of cryptological strength is inherently fuzzy

Not really. The OTP system is proveably secure.

>and not entirely separable from subjectivity and concepts like
>confidence intervals, i.e. no security can be claimed on an absolute
>scale in practice. But experts might refute my un-knowledgeable
>assertions.

If someone tells you that he can demonstrate that a given number is crypto-grade random without considering the way it is generated, he is making a fundamental error, one of the most widespread errors in cryptography.

Bob Knauer

"An honest man can feel no pleasure in the exercise of power over his fellow citizens."
--Thomas Jefferson

Subject: Re: Random numbers from a sound card?
Date: Tue, 26 Jan 1999 14:19:59 +0100
From: Mok-Kong Shen <mok-kong.shen@stud.uni-muenchen.de>
Message-ID: <36ADC0FF.68F55692@stud.uni-muenchen.de>
References: <36ad09fc.46447347@nntp.ix.netcom.com>
Newsgroups: sci.crypt
Lines: 14

R. Knauer wrote:

> >I guess the issue of cryptological strength is inherently fuzzy
>
> Not really. The OTP system is proveably secure.

Once again I assert that this is a (for all practical purposes) useless fact, because OTP presuppose (absolutely) true randomness and there is no way of determining that in practice. I suppose (with my meager knowledge of physics) this is almost the same as saying at at at 0 Kelvin you can halt the motions of all atoms (but you can't get to 0 Kelvin, only very close to it).

M. K. Shen

Subject: Re: Random numbers from a sound card?
Date: Tue, 26 Jan 1999 19:11:42 GMT
From: rcktexas@ix.netcom.com (R. Knauer)
Message-ID: <36ae0cb6.27338280@nntp.ix.netcom.com>
References: <36ADC0FF.68F55692@stud.uni-muenchen.de>
Newsgroups: sci.crypt
Lines: 68

On Tue, 26 Jan 1999 14:19:59 +0100, Mok-Kong Shen
<mok-kong.shen@stud.uni-muenchen.de> wrote:

>> Not really. The OTP system is proveably secure.

>Once again I assert that this is a (for all practical purposes)
>useless fact,

Sorry, but that's nonsense.

For all practical purposes the OTP is proveably secure. That means that you can build an OTP system that is secure to within a level of precision can be made arbitrarily small.

To do that you must build a physical device which can generate all possible sequences of a given finite length equiprobably. That is possible using quantum mechanical processes and good electronic design.

The hot line between Washington and Moscow is (supposedly) protected by an OTP. Conversations on that line can be tapped and interfered with in principle by anyone close enough to the equipment. Do you think the two most dangerous govts in the world would trust the fate of Planet Earth to an insecure communications link? Hardly.

Nothing we humans build is Perfect, but we are able to build things that are very damn close to Perfect. We can build TRNGs that are perfect enough to send messages which would take more energy to analyze than is available in the Universe.

How much more Perfect do you want, even in a practical sense?

>because OTP presuppose (absolutely) true randomness

>and there is no way of determining that in practice.

Sure there is. Just look at how the numbers are being generated. That will tell you if they are random.

>I suppose
>(with my meager knowledge of physics) this is almost the same as
>saying at at at 0 Kelvin you can halt the motions of all atoms

You are not aware of the so-called "zero point" vacuum fluctuations which persist even at 0 Kelvin. If all motion stopped at 0 Kelvin, the Universe would cease to exist - no photons, no particles, no forces - nothing.

>(but you can't get to 0 Kelvin, only very close to it).

You can get exceedingly close to it, like one milli-degree close to it. That's one thousandth of a degree close to it. How much closer would you want to get to be closer than very close to it?

Is calculus impossible because numbers can never actually reach the limit required to calculate a derivative or an integral? People in the 17th century, when Newton and Leibnitz invented calculus, thought calculus was wrong because those limits could never be reached in a "practical" sense. Yet calculus went on being correct despite them.

And crypto-grade randomness goes on being correct in a very practical sense too, despite the lack of perfection in a practical sense.

Bob Knauer

"An honest man can feel no pleasure in the exercise of power over his fellow citizens."
--Thomas Jefferson

Subject: Re: Random numbers from a sound card?
Date: Wed, 27 Jan 1999 14:56:34 +0100
From: Mok-Kong Shen <mok-kong.shen@stud.uni-muenchen.de>
Message-ID: <36AF1B12.FBD87AB5@stud.uni-muenchen.de>
References: <36ae0cb6.27338280@nntp.ix.netcom.com>
Newsgroups: sci.crypt
Lines: 28

R. Knauer wrote:
>

> Nothing we humans build is Perfect, but we are able to build things
> that are very damn close to Perfect. We can build TRNGs that are
> perfect enough to send messages which would take more energy to
> analyze than is available in the Universe.
>
> How much more Perfect do you want, even in a practical sense?

I am not against having something ideal and perfect as a standard for approximation (to be strived at in practice) or for pedagogical purpose. But to say there IS (in the sence of EXISTS) something perfect can be misleading.

>
> >because OTP presuppose (absolutely) true randomness
> >and there is no way of determining that in practice.
>
> Sure there is. Just look at how the numbers are being generated. That
> will tell you if they are random.

To 'just look' is certainly not ensuring (compare watching a magician pulling rabits out of his hat). We have to ascertain how 'random' the sequence we get really is. And that's one of the real and big problem for the practice.

M. K. Shen

Subject: Re: Random numbers from a sound card?
Date: Wed, 27 Jan 1999 15:02:36 GMT
From: rcktexas@ix.netcom.com (R. Knauer)
Message-ID: <36af2a34.18209443@nntp.ix.netcom.com>
References: <36AF1B12.FBD87AB5@stud.uni-muenchen.de>
Newsgroups: sci.crypt
Lines: 17

On Wed, 27 Jan 1999 14:56:34 +0100, Mok-Kong Shen
<mok-kong.shen@stud.uni-muenchen.de> wrote:

>But to say there IS (in the sense of EXISTS) something
>perfect can be misleading.

Does a Perfect Circle EXIST?

If you say is does, is that misleading?

Bob Knauer

"No Freeman shall ever be debarred the use of arms. The strongest
reason for the people to retain the right to keep and bear arms is,
as a last resort, to protect themselves against tyranny in government."
--Thomas Jefferson

Subject: Re: Random numbers from a sound card?
Date: Wed, 27 Jan 1999 16:44:40 +0100
From: Mok-Kong Shen <mok-kong.shen@stud.uni-muenchen.de>
Message-ID: <36AF3468.78EDE075@stud.uni-muenchen.de>
References: <36af2a34.18209443@nntp.ix.netcom.com>
Newsgroups: sci.crypt
Lines: 16

R. Knauer wrote:
>
> On Wed, 27 Jan 1999 14:56:34 +0100, Mok-Kong Shen
> <mok-kong.shen@stud.uni-muenchen.de> wrote:
>
> >But to say there IS (in the sense of EXISTS) something
> >perfect can be misleading.
>
> Does a Perfect Circle EXIST?
>
> If you say is does, is that misleading?

If the word 'IS' is employed in a context without the connotation
of 'EXISTS' then it is NOT misleading, otherwise it IS misleading.

M. K. Shen

Subject: Re: Random numbers from a sound card?
Date: Wed, 27 Jan 1999 16:15:17 GMT
From: rcktexas@ix.netcom.com (R. Knauer)
Message-ID: <36af3b4d.22586607@nntp.ix.netcom.com>
References: <36AF3468.78EDE075@stud.uni-muenchen.de>
Newsgroups: sci.crypt
Lines: 20

On Wed, 27 Jan 1999 16:44:40 +0100, Mok-Kong Shen
<mok-kong.shen@stud.uni-muenchen.de> wrote:

>If the word 'IS' is employed in a context without the connotation
>of 'EXISTS' then it is NOT misleading, otherwise it IS misleading.

You are beginning to sound just like Bill Clinton:

"It all depends on what the meaning of the word 'is' is."

<jeez>

Bob Knauer

"No Freeman shall ever be debarred the use of arms. The strongest
reason for the people to retain the right to keep and bear arms is,
as a last resort, to protect themselves against tyranny in government."
--Thomas Jefferson

Subject: Re: Random numbers from a sound card?
Date: Wed, 27 Jan 1999 17:46:44 +0100
From: Mok-Kong Shen <mok-kong.shen@stud.uni-muenchen.de>
Message-ID: <36AF42F4.5B9DC5CD@stud.uni-muenchen.de>
References: <36af3b4d.22586607@nntp.ix.netcom.com>
Newsgroups: sci.crypt
Lines: 23

R. Knauer wrote:
>
> On Wed, 27 Jan 1999 16:44:40 +0100, Mok-Kong Shen
> <mok-kong.shen@stud.uni-muenchen.de> wrote:
>
> >If the word 'IS' is employed in a context without the connotation
> >of 'EXISTS' then it is NOT misleading, otherwise it IS misleading.
>
> You are beginning to sound just like Bill Clinton:
>
> "It all depends on what the meaning of the word 'is' is."

That way clearly stated in my previous post, quoted below:

   But to say there IS (in the sense of EXISTS) something
   perfect can be misleading.

A word can have a multitude of meanings. I was prudent enough
to put the parentheses above to make sure that there could be
no misunderstanding. I regret that my attempt was appraently
not successful.

M. K. Shen

Subject: Re: Random numbers from a sound card?
Date: Wed, 27 Jan 1999 17:36:54 GMT
From: rcktexas@ix.netcom.com (R. Knauer)
Message-ID: <36af4e20.27405937@nntp.ix.netcom.com>
References: <36AF42F4.5B9DC5CD@stud.uni-muenchen.de>
Newsgroups: sci.crypt
Lines: 34

On Wed, 27 Jan 1999 17:46:44 +0100, Mok-Kong Shen
<mok-kong.shen@stud.uni-muenchen.de> wrote:

>> "It all depends on what the meaning of the word 'is' is."

>That way clearly stated in my previous post, quoted below:

> But to say there IS (in the sense of EXISTS) something
> perfect can be misleading.

>A word can have a multitude of meanings. I was prudent enough
>to put the parentheses above to make sure that there could be
>no misunderstanding. I regret that my attempt was appraently
>not successful.

You must be a mathematician.

As Greg Chaitin says in his latest book. "The Unknowable", physicists
have a sense of humor (BTW, I am a physicist), but mathematicians do
not have a sense of humor.

Which is not completely true because Chaitin is a mathematician and he
has a sense of humor.

Whatever.

Bob Knauer

"No Freeman shall ever be debarred the use of arms. The strongest
reason for the people to retain the right to keep and bear arms is,
as a last resort, to protect themselves against tyranny in government."
--Thomas Jefferson

Subject: Re: Random numbers from a sound card?
Date: Wed, 27 Jan 1999 12:02:38 -0600
From: Medical Electronics Lab <rosing@physiology.wisc.edu>
Message-ID: <36AF54BE.5771@physiology.wisc.edu>
References: <36AF1B12.FBD87AB5@stud.uni-muenchen.de>
Newsgroups: sci.crypt
Lines: 23

Mok-Kong Shen wrote:
> I am not against having something ideal and perfect as a standard
> for approximation (to be strived at in practice) or for pedagogical
> purpose. But to say there IS (in the sence of EXISTS) something
> perfect can be misleading.

Kind of depends on how you define "perfect". Perfect for what and
measured in what way? We can certainly build a TRNG which is
perfect in any measureable sense.

> To 'just look' is certainly not ensuring (compare watching a
> magician pulling rabits out of his hat). We have to ascertain
> how 'random' the sequence we get really is. And that's one of
> the real and big problem for the practice.

Which is what makes this whole discussion so much fun. DIEHARD
and Diaphony and autocorrelation all measure "random" in a slightly
different way. If the output of a TRNG appears random to all those
tests, we can say it "looks" random. It is "perfect" as far
as we can measure. Isn't that good enough?

Patience, persistence, truth,
Dr. mike

Subject: Re: Random numbers from a sound card?
Date: Wed, 27 Jan 1999 12:06:35 -0700
From: "Tony T. Warnock" <u091889@cic-mail.lanl.gov>
Message-ID: <36AF63BB.18D3EEA9@cic-mail.lanl.gov>
References: <36AF54BE.5771@physiology.wisc.edu>
Newsgroups: sci.crypt
Lines: 46

Medical Electronics Lab wrote:

> Mok-Kong Shen wrote:
> > I am not against having something ideal and perfect as a standard

> > for approximation (to be strived at in practice) or for pedagogical
> > purpose. But to say there IS (in the sence of EXISTS) something
> > perfect can be misleading.
>
> Kind of depends on how you define "perfect". Perfect for what and
> measured in what way? We can certainly build a TRNG which is
> perfect in any measureable sense.
>
> > To 'just look' is certainly not ensuring (compare watching a
> > magician pulling rabits out of his hat). We have to ascertain
> > how 'random' the sequence we get really is. And that's one of
> > the real and big problem for the practice.
>
> Which is what makes this whole discussion so much fun. DIEHARD
> and Diaphony and autocorrelation all measure "random" in a slightly
> different way. If the output of a TRNG appears random to all those
> tests, we can say it "looks" random. It is "perfect" as far
> as we can measure. Isn't that good enough?
>
> Patience, persistence, truth,
> Dr. mike

It's not clear what is wanted here. In limit (infinitely long sequences) both the complexity based and the frequency (statistical) based definitions of random are equivalent (per Martin Lof). For finite sequences (actually for computable sequences, IMHO) these are not necessarily equivalent. It is easy to produce sequences that satisfy the strong law of large numbers. Champernowne's sequence comes to mind: 01,1011,100101110111,.... It is not very complex computationally. It does have the proper frequency of everything, that is, each k bit sequence has limiting frequence 1/2^k. Unfortunately I do not know of any easily constructed sequence that satisfy the law of the iterated logarithm. I do not even know how to test for this. It would be a requirement for a "statistically random" sequence.

It's possible that one can only list a set of criteria and check if your sequence satisfy them. Again, most of these criteria are not computable but "almost all" sequences satisfy them

Tony

Subject: Re: Random numbers from a sound card?
Date: Wed, 27 Jan 1999 23:01:14 GMT
From: rcktexas@ix.netcom.com (R. Knauer)
Message-ID: <36af98b1.46494265@nntp.ix.netcom.com>
References: <36AF54BE.5771@physiology.wisc.edu>
Newsgroups: sci.crypt
Lines: 50

On Wed, 27 Jan 1999 12:02:38 -0600, Medical Electronics Lab
<rosing@physiology.wisc.edu> wrote:

>> To 'just look' is certainly not ensuring (compare watching a
>> magician pulling rabits out of his hat). We have to ascertain
>> how 'random' the sequence we get really is. And that's one of
>> the real and big problem for the practice.

>Which is what makes this whole discussion so much fun.

Then you're a masochist. :-)

Once you catch on to all this, you will see why.

>DIEHARD
>and Diaphony and autocorrelation all measure "random" in a slightly
>different way.

Those things don't measure the crypto-grade randomness of finite numbers at all. They try to make inferences about the generator from finite samples, which is useless for purposes of crypto. They will pass the outputs of PRNGs that can be cracked.

We need an update to the Snake Oil FAQ desperately!

>If the output of a TRNG appears random to all those
>tests, we can say it "looks" random.

Just what makes a finite number produced by a TRNG "look random"?

Why do you thing that characteristics that apply only to infinite numbers can also apply to finite ones with equal certitude?

What does "vanishingly small" mean to you?

>It is "perfect" as far as we can measure.

That measure is worthless for crypto-grade random numbers.

> Isn't that good enough?

Nope. Not even close.

Bob Knauer

"No Freeman shall ever be debarred the use of arms. The strongest
reason for the people to retain the right to keep and bear arms is,
as a last resort, to protect themselves against tyranny in government."
--Thomas Jefferson

Subject: Re: Random numbers from a sound card?
Date: Thu, 28 Jan 1999 12:47:39 -0600
From: Medical Electronics Lab <rosing@physiology.wisc.edu>
Message-ID: <36B0B0CB.1974@physiology.wisc.edu>
References: <36af98b1.46494265@nntp.ix.netcom.com>
Newsgroups: sci.crypt
Lines: 56

R. Knauer wrote:
>
> Then you're a masochist. :-)

:-) I think you're right about that!

> Once you catch on to all this, you will see why.

It's pretty clear we have different opinions. The best I can
hope for is more descriptions so I can find out what the core
assumption is that we disagree on. Neither one of us will change
:-)

> Those things don't measure the crypto-grade randomness of finite
> numbers at all. They try to make inferences about the generator from
> finite samples, which is useless for purposes of crypto. They will
> pass the outputs of PRNGs that can be cracked.

So you need an infinite sequence of bits to prove that something
is crypto-grade random, yes?

> Just what makes a finite number produced by a TRNG "look random"?

Is a 10 megabyte block of random bits a single number? Or is it
80 million individual numbers? For the latter case, it looks
random if it can pass all the tests for randomness that
mathematicians have dreamed up. In the former case, if it isn't
printable ascii, then it will probably look random no matter
what.

> Why do you thing that characteristics that apply only to infinite
> numbers can also apply to finite ones with equal certitude?

What characteristics are you talking about? Integrals over a
finite range and binomial or poisson distributions are all based
on finite samples. All the DIEHARD tests are based on finite
samples. I am assuming that Marsaglia knows what he's doing,
but maybe you can correct him?

> What does "vanishingly small" mean to you?

Less than I can measure.

> That measure is worthless for crypto-grade random numbers.

Yes, well, expand on "crypto-grade" a bit.

> > Isn't that good enough?
>
> Nope. Not even close.

:-) See, I told you we disagree. Let's keep it that way,
makes for a nice long discussion.

Patience, persistence, truth,
Dr. mike

Subject: Re: Random numbers from a sound card?
Date: Thu, 28 Jan 1999 23:07:11 GMT
From: rcktexas@ix.netcom.com (R. Knauer)
Message-ID: <36b0d008.1959958@nntp.ix.netcom.com>
References: <36B0B0CB.1974@physiology.wisc.edu>
Newsgroups: sci.crypt
Lines: 111

On Thu, 28 Jan 1999 12:47:39 -0600, Medical Electronics Lab
<rosing@physiology.wisc.edu> wrote:

>It's pretty clear we have different opinions. The best I can
>hope for is more descriptions so I can find out what the core
>assumption is that we disagree on. Neither one of us will change

You will change once you catch on. I did.

A year ago I came onto sci.crypt with ill-formed notions of

crypto-grade randomness. After what seemed like a thousand posts from many participants, the truth emerged.

I have capsulized that truth several times recently but some people, including you, still have not caught on. When you do catch on, you will look back and wonder how you could have been so confused about such a straightforward concept. I did.

>So you need an infinite sequence of bits to prove that something
>is crypto-grade random, yes?

You cannot prove the crypto-grade randomness of a finite number algorithmically. You can for an infinite number, but that is useless.

The only way you can prove the crypto-grade randomness of a finite number is to consider the method of generation. If the generator is a TRNG, as we have defined it here several times recently, then the numbers it generates are crypto-grade random numbers.

>Is a 10 megabyte block of random bits a single number?

Yes.

>Or is it 80 million individual numbers?

Yes.

>For the latter case, it looks
>random if it can pass all the tests for randomness that
>mathematicians have dreamed up.

Wrong. You might be able to infer some things about the numbers that fool you into thinking they are random, but that does not make them crypto-random.

Keep in mind that many PRNGs pass statistical tests.

>In the former case, if it isn't
>printable ascii, then it will probably look random no matter
>what.

Numbers don't "look" crypto-random.

The number 1111111111 is a crypto-grade random number, because it was generated by a TRNG. Or, may it is not because it was not generated by a TRNG. You cannot tell unless you know the generation process.

Tell me if you think 111111111 is crypto-grade random or not.

>> Why do you thing that characteristics that apply only to infinite
>> numbers can also apply to finite ones with equal certitude?

>What characteristics are you talking about?

The characteristic of randomness. Infinite numbers have characteristics which can be related to randomness. If an infinite number is a normal number, it is random. Finite numbers cannot be normal numbers - they are not big enough.

For example, if you can prove that pi is a normal number, then it is a random number.

>Integrals over a
>finite range and binomial or poisson distributions are all based
>on finite samples.

Do they measure crypto-grade randomness of finite numbers?

If they could, these algorithms you propose could also be used to solve Godel's incompleteness problem, Turing's halting problem and Chaitin's complexity problem.

>All the DIEHARD tests are based on finite
>samples. I am assuming that Marsaglia knows what he's doing,
>but maybe you can correct him?

You correct him, when you discover the truth.

>> What does "vanishingly small" mean to you?

>Less than I can measure.

Explain your method of measurement.

>Yes, well, expand on "crypto-grade" a bit.

Proveably secure when used with the OTP cryptosystem.

>:-) See, I told you we disagree. Let's keep it that way,
>makes for a nice long discussion.

Last time it was over 1,000 posts. I am beginning to think I was the only one who got anything out of them.

Bob Knauer

"No Freeman shall ever be debarred the use of arms. The strongest
reason for the people to retain the right to keep and bear arms is,
as a last resort, to protect themselves against tyranny in government."
--Thomas Jefferson

Subject: Re: Random numbers from a sound card?
Date: Fri, 29 Jan 1999 11:47:15 +0100
From: Mok-Kong Shen <mok-kong.shen@stud.uni-muenchen.de>
Message-ID: <36B191B3.89CCEFA3@stud.uni-muenchen.de>
References: <36b0d008.1959958@nntp.ix.netcom.com>
Newsgroups: sci.crypt
Lines: 19

R. Knauer wrote:
>

>
> You cannot prove the crypto-grade randomness of a finite number
> algorithmically. You can for an infinite number, but that is useless.
>
> The only way you can prove the crypto-grade randomness of a finite
> number is to consider the method of generation. If the generator is a
> TRNG, as we have defined it here several times recently, then the
> numbers it generates are crypto-grade random numbers.

Ah! Finally one knows exactly what the term 'crypto-grade random
numbers' you employ means: These are DEFINED to be the output
from a hardware generator. If follows obviously then that there
is NO need whatsoever of testing the sequences obtained, since they
are BY DEFINITION 'crypto-grade'!

M. K. Shen

Subject: Re: Random numbers from a sound card?
Date: Fri, 29 Jan 1999 13:37:25 GMT
From: rcktexas@ix.netcom.com (R. Knauer)
Message-ID: <36b1b932.2360123@nntp.ix.netcom.com>
References: <36B191B3.89CCEFA3@stud.uni-muenchen.de>
Newsgroups: sci.crypt
Lines: 26

On Fri, 29 Jan 1999 11:47:15 +0100, Mok-Kong Shen
<mok-kong.shen@stud.uni-muenchen.de> wrote:

>> The only way you can prove the crypto-grade randomness of a finite
>> number is to consider the method of generation. If the generator is a
>> TRNG, as we have defined it here several times recently, then the
>> numbers it generates are crypto-grade random numbers.

>Ah! Finally one knows exactly what the term 'crypto-grade random
>numbers' you employ means: These are DEFINED to be the output
>from a hardware generator. If follows obviously then that there
>is NO need whatsoever of testing the sequences obtained, since they
>are BY DEFINITION 'crypto-grade'!

Are you being deliberatly obtuse - or does it come naturally?

Nothing that you said above follows from what I said. A TRNG is not a
True Random Number Generator just because it is a hardware device.

Bob Knauer

"No Freeman shall ever be debarred the use of arms. The strongest
reason for the people to retain the right to keep and bear arms is,
as a last resort, to protect themselves against tyranny in government."
--Thomas Jefferson

Subject: Re: Random numbers from a sound card?
Date: 29 Jan 1999 09:14:11 -0500
From: juola@mathcs.duq.edu (Patrick Juola)
Message-ID: <78sfnjouo1@quine.mathcs.duq.edu>
References: <36B191B3.89CCEFA3@stud.uni-muenchen.de>
Newsgroups: sci.crypt
Lines: 21

In article <36B191B3.89CCEFA3@stud.uni-muenchen.de>,
Mok-Kong Shen <mok-kong.shen@stud.uni-muenchen.de> wrote:
>R. Knauer wrote:
>>
>
>>
>> You cannot prove the crypto-grade randomness of a finite number
>> algorithmically. You can for an infinite number, but that is useless.
>>
>> The only way you can prove the crypto-grade randomness of a finite
>> number is to consider the method of generation. If the generator is a
>> TRNG, as we have defined it here several times recently, then the

>> numbers it generates are crypto-grade random numbers.
>
>Ah! Finally one knows exactly what the term 'crypto-grade random
>numbers' you employ means: These are DEFINED to be the output
>from a hardware generator.

No. Not all hardware generators are TRNG.

-kitten

Subject: Re: Random numbers from a sound card?
Date: Fri, 29 Jan 1999 08:38:00 -0700
From: "Tony T. Warnock" <u091889@cic-mail.lanl.gov>
Message-ID: <36B1D5D8.9E4A7B55@cic-mail.lanl.gov>
References: <36b0d008.1959958@nntp.ix.netcom.com>
Newsgroups: sci.crypt
Lines: 35

R. Knauer wrote:

> The characteristic of randomness. Infinite numbers have
> characteristics which can be related to randomness. If an infinite
> number is a normal number, it is random. Finite numbers cannot be
> normal numbers - they are not big enough.
>
> For example, if you can prove that pi is a normal number, then it is a
> random number.

Normality is certainly necessary but not sufficient. It's a good start.

More than normality is needed. I can give you many normal numbers but none of them are "random." Champernowne's number is the simplest example: 1,10,11,100,101,110,111,...=11011100101110111.... It is easy to prove that all k-bit patterns have the proper frequency. This is all that is needed for normality. (The concept of normality was introduced by Borel about 1909.) The digits of a normal number satisfy the strong law of large numbers, that is, 1/2 ones, 1/2 zeros, 1/4 00's, 1/4 01's, 1/4 10's, 1/4 11's, ..., 1/1024 1101101101's, etc.

The problem is that the strong law of large numbers is not very strong. In Champernowne's number, the excess of ones over zeros grows as N/log(N) for N bits. The ratio goes like 1/2+1/log(N), really slow. The dispersion is also not correct. The law of the iterated logarithm fails for all these sequences.

Of course both the above laws (large numbers, iterated logarithm) are statistical in nature and do not indicate how difficult it is to guess successive bits of a number. Complexity of computation and statistical properties are only equivalent in the limit of infnitely many infinitely long sequences.

Tony

Subject: Re: Random numbers from a sound card?
Date: Sat, 30 Jan 1999 03:09:39 GMT
From: rcktexas@ix.netcom.com (R. Knauer)
Message-ID: <36b272f8.49918378@nntp.ix.netcom.com>
References: <36B1D5D8.9E4A7B55@cic-mail.lanl.gov>
Newsgroups: sci.crypt
Lines: 70

On Fri, 29 Jan 1999 08:38:00 -0700, "Tony T. Warnock"
<u091889@cic-mail.lanl.gov> wrote:

>Normality is certainly necessary but not sufficient. It's a good start.

>More than normality is needed. I can give you many normal numbers but none
>of them are "random."

Are they infinite?

>Champernowne's number is the simplest example:
>1,10,11,100,101,110,111,...=11011100101110111.... It is easy to prove that
>all k-bit patterns have the proper frequency.

Yes, but only if the number is infinite.

>This is all that is needed
>for normality. (The concept of normality was introduced by Borel about
>1909.) The digits of a normal number satisfy the strong law of large
>numbers, that is, 1/2 ones, 1/2 zeros, 1/4 00's, 1/4 01's, 1/4 10's, 1/4
>11's, ..., 1/1024 1101101101's, etc.

Chaitin cover this in his papers - for those who want an accessible reference.

>The problem is that the strong law of large numbers is not very strong. In
>Champernowne's number, the excess of ones over zeros grows as N/log(N) for
>N bits.

Is that really a problem? Whoevewr said that bias was an intrinsic

property of infinite random numbers?

>The ratio goes like 1/2+1/log(N), really slow. The dispersion is
>also not correct. The law of the iterated logarithm fails for all these
>sequences.

This what I like about the Internet in general, and Usenet forums like sci.crypt in particular. There is always someone who knows the something about something - someone who is willing to jump in and expose that.

Without the Truth to seek out, life is completely meaningless. [Cf. Camus, "The Myth Of Sysiphus" and the concept of "Lucidity".]

Your further elaborations would be most higly regarded by me amd all the lurkers on sci.crypt. The concept of randomness is fundamental to an understanding of how we consider Order, the thing which distinguishes us from dirt. The concept of randomness is at the heart of Quantum Mechanics, which has incredible predictive value.

>Of course both the above laws (large numbers, iterated logarithm) are
>statistical in nature and do not indicate how difficult it is to guess
>successive bits of a number. Complexity of computation and statistical
>properties are only equivalent in the limit of infnitely many infinitely
>long sequences.

Another excellent contribution to the FAQ on crypto-grade randomness.

But I point out that computational complexity has nothing fundamental to do with crypto-grade randomness, nor QM. In those realms everything is possible, even the most simple of sequences. In fact, I believe we are here because the simpler sequences prevailed.

Bob Knauer

"No Freeman shall ever be debarred the use of arms. The strongest reason for the people to retain the right to keep and bear arms is, as a last resort, to protect themselves against tyranny in government."
--Thomas Jefferson

Subject: Re: Random numbers from a sound card?
Date: Sat, 30 Jan 1999 14:51:49 GMT
From: rcktexas@ix.netcom.com (R. Knauer)
Message-ID: <36b31a67.8855773@nntp.ix.netcom.com>
References: <36B1D5D8.9E4A7B55@cic-mail.lanl.gov>
Newsgroups: sci.crypt
Lines: 38

On Fri, 29 Jan 1999 08:38:00 -0700, "Tony T. Warnock"
<u091889@cic-mail.lanl.gov> wrote:

>Champernowne's number is the simplest example:
>1,10,11,100,101,110,111,...=11011100101110111.... It is easy to prove that
>all k-bit patterns have the proper frequency. This is all that is needed
>for normality. (The concept of normality was introduced by Borel about
>1909.) The digits of a normal number satisfy the strong law of large
>numbers, that is, 1/2 ones, 1/2 zeros, 1/4 00's, 1/4 01's, 1/4 10's, 1/4
>11's, ..., 1/1024 1101101101's, etc.

>The problem is that the strong law of large numbers is not very strong. In
>Champernowne's number, the excess of ones over zeros grows as N/log(N) for
>N bits. The ratio goes like 1/2+1/log(N), really slow. The dispersion is
>also not correct. The law of the iterated logarithm fails for all these
>sequences.

In re-reading this I spotted something I do not understand. You state that for the Champernowne number "all k-bit patterns have the proper frequency". I assume that is true for k = 1, one of the possible values for k.

Then you say that in Champernowne's number there is an "excess of ones over zeros". How can that be if "all k-bit patterns have the proper frequency"? The "proper frequency" for k = 1 as described by you: "The digits of a normal number satisfy the strong law of large numbers, that is, 1/2 ones, 1/2 zeros".

How come you state that Champernowne's number has an "excess of ones over zeros"?

Bob Knauer

"No Freeman shall ever be debarred the use of arms. The strongest reason for the people to retain the right to keep and bear arms is, as a last resort, to protect themselves against tyranny in government."
--Thomas Jefferson

Subject: Re: Random numbers from a sound card?
Date: Sat, 30 Jan 1999 14:31:17 -0500
From: "Trevor Jackson, III" <fullmoon@aspi.net>
Message-ID: <36B35E04.3DF9ED19@aspi.net>

References: <36b31a67.8855773@nntp.ix.netcom.com>
Newsgroups: sci.crypt
Lines: 46

R. Knauer wrote:

> On Fri, 29 Jan 1999 08:38:00 -0700, "Tony T. Warnock"
> <u091889@cic-mail.lanl.gov> wrote:
>
> >Champernowne's number is the simplest example:
> >1,10,11,100,101,110,111,...=11011100101110111.... It is easy to prove that
> >all k-bit patterns have the proper frequency. This is all that is needed
> >for normality. (The concept of normality was introduced by Borel about
> >1909.) The digits of a normal number satisfy the strong law of large
> >numbers, that is, 1/2 ones, 1/2 zeros, 1/4 00's, 1/4 01's, 1/4 10's, 1/4
> >11's, ..., 1/1024 1101101101's, etc.
>
> >The problem is that the strong law of large numbers is not very strong. In
> >Champernowne's number, the excess of ones over zeros grows as N/log(N) for
> >N bits. The ratio goes like 1/2+1/log(N), really slow. The dispersion is
> >also not correct. The law of the iterated logarithm fails for all these
> >sequences.
>
> In re-reading this I spotted something I do not understand. You state
> that for the Champernowne number "all k-bit patterns have the proper
> frequency". I assume that is true for k = 1, one of the possible
> values for k.
>
> Then you say that in Champernowne's number there is an "excess of ones
> over zeros". How can that be if "all k-bit patterns have the proper
> frequency"? The "proper frequency" for k = 1 as described by you: "The
> digits of a normal number satisfy the strong law of large numbers,
> that is, 1/2 ones, 1/2 zeros".
>
> How come you state that Champernowne's number has an "excess of ones
> over zeros"?

Because all the leading zeros are suppressed.

>
>
> Bob Knauer
>
> "No Freeman shall ever be debarred the use of arms. The strongest
> reason for the people to retain the right to keep and bear arms is,
> as a last resort, to protect themselves against tyranny in government."
> --Thomas Jefferson

Subject: Re: Random numbers from a sound card?
Date: Sun, 31 Jan 1999 00:06:40 GMT
From: rcktexas@ix.netcom.com (R. Knauer)
Message-ID: <36b39e73.42659320@nntp.ix.netcom.com>
References: <36B35E04.3DF9ED19@aspi.net>
Newsgroups: sci.crypt
Lines: 17

On Sat, 30 Jan 1999 14:31:17 -0500, "Trevor Jackson, III"
<fullmoon@aspi.net> wrote:

>> How come you state that Champernowne's number has an "excess of ones
>> over zeros"?

>Because all the leading zeros are suppressed.

Can you elaborate with an example.

Bob Knauer

"I place economy among the first and most important virtues and
public debt as the greatest dangers to be feared. We must not
let our rulers load us with perpetual debt."
--Thomas Jefferson

Subject: Re: Random numbers from a sound card?
Date: 26 Jan 1999 14:59:26 -0500
From: juola@mathcs.duq.edu (Patrick Juola)
Message-ID: <78l6qu$k97$1@quine.mathcs.duq.edu>
References: <36ADC0FF.68F55692@stud.uni-muenchen.de>
Newsgroups: sci.crypt
Lines: 25

In article <36ADC0FF.68F55692@stud.uni-muenchen.de>,
Mok-Kong Shen <mok-kong.shen@stud.uni-muenchen.de> wrote:
>R. Knauer wrote:
>
>> >I guess the issue of cryptological strength is inherently fuzzy
>>
>> Not really. The OTP system is proveably secure.

>
>Once again I assert that this is a (for all practical purposes)
>useless fact, because OTP presuppose (absolutely) true randomness
>and there is no way of determining that in practice.

Not quite. The randomness that the OTP presumes works out to
be exactly the same problem as key generation for a key system --
if (somehow) the attacker can predict which key you are going
to use, then the attacker can unbutton your messages more or
less at will. Similarly if the attacker can force you to use
a particular key.

So, yes, you'll probably not build a "perfect" OTP in practice,
any more than you'll be able to get a bug-free computer program.
That doesn't mean that there aren't techniques that are *less*
likely to approach perfection than others.

-kitten

Subject: Re: Random numbers from a sound card?
Date: Tue, 26 Jan 1999 18:15:11 +0000
From: pla@sktb.demon.co.uk (Paul L. Allen)
Message-ID: <f8BSgej030n@sktb.demon.co.uk>
References: <36ad09fc.46447347@nntp.ix.netcom.com>
Newsgroups: sci.crypt
Lines: 38

In article <36ad09fc.46447347@nntp.ix.netcom.com>
    rcktexas@ix.netcom.com (R. Knauer) writes:

> On Mon, 25 Jan 1999 20:11:33 +0100, Mok-Kong Shen
> <mok-kong.shen@stud.uni-muenchen.de> wrote:
>
> >>   How would you test the 'quality' of the generated random number
> >> stream?
>
> >There are tests for statistical quality, e.g. Maurer's universal
> >statistical test. I am ignorant of tests for crypto quality.
>
> That's because there aren't any.
>
> It is a fundamental precept of crytpography that randomness is a
> property of how a number is generated, not a property of a number
> itself.

I'd be *extremely* worried about a sound card (particularly with little
in the way of input) picking up mains power hum and radiated noise from
signal lines in the computer. Obviously they'll only affect a few of
the lower bits of output (or you'd hear it when you played sounds back)
but that might be enough to weaken the randomness enough to cause problems.
Boiling down the entropy with a cryptographic hash function probably
gets rid of it, but if you have a crappy sound card you may have to do
a lot more boiling away than with a good one.

Noise diodes strike me as being safer. The fundamental mechanism of
noise generation in them may well be chaotic and I've seen worries that
chaotic loci can be close to periodic for long periods of time, but there
are likely to be many individual sources of chaotic noise in the diode
(due to random distribution of dopant atoms) which may make things alot
safer. Probably. It really needs somebody whose done detailed work on
those noise mechanisms to comment.

--Paul

Subject: Re: Random numbers from a sound card?
Date: Tue, 26 Jan 1999 20:12:38 GMT
From: randombit@my-dejanews.com
Message-ID: <78l7je$cc6$1@nnrp1.dejanews.com>
References: <36acb8b1.5374650@news.willapabay.org>
Newsgroups: sci.crypt
Lines: 79

In article <36acb8b1.5374650@news.willapabay.org>,
  ross@hypertools.com (David Ross) wrote:
>   Has anyone had success using a sound card (like a Sound Blaster) to
> generate streams of random numbers?

Yes.

>   What sort of audio source would you suspect would be the best to use
> in generating random numbers?

I used an old radio shack mono fm radio, with antenna removed,
tuned to hiss, at high volume, fed into the sound card.

Later I got a video/radio digitizer, which I can tune to
FM hiss, which is more self contained.

This produces an apparently uniformly distributed noise spectrum,
using a pc-based spectrum analyzer.

But this doesn't have full entropy, and you have to distill (see
RFC 1750) the bits. I experimented with parity-of-N bits,
and used Maurer's Universal statistical test for RNGs to measure
the entropy. When you distill enough, the entropy reaches its
expected value.

Some people might recommend a strong hashing function (e.g., a thousand
raw bits hashed with MD5 down to a fixed output size). This is
complex and I found unnecessary; simple parity works, though it
may waste more bits than a serious hash would. But bits are cheap,
and xor is fast.

> How would you test the 'quality' of the generated random number
> stream?

1. Marsaglia's Diehard suite of statistical (structure) tests.
This suite goes far beyond the FIPS suggestions.

2. Maurer's Universal statistical test, which approximates
the entropy of a sample using a formally motivated,
compression-like algorithm.

As "calibration standards" I used the "RAND million normal digits and their
deviant friends", and also block ciphers run in feedback modes (ie,
as PRNGs).

I've also got a parallel-port compatible geiger counter and a
microcurie of americium, but i haven't careful studies on these yet.
But they are cool toys :-)

You will learn that you *always* have to distill raw bits.
And you may observe that very few hardware RNGs actually monitor their output
quality (especially on-line), though it seems to me you should.

Also note that a 'loud' source of hiss is preferable. Were I using
an acoustic microphone as my raw input, I would locate it next
to the frother on my espresso machine and blow steam out of it,
rather than counting on the wind or ambient brownian effects.

Note that even using a highly structured signal (e.g., digitized
video program including your local receiver noise) you could generate
good bits, but you'd have to distill bushels of them.

Have fun,

randombit

----------== Posted via Deja News, The Discussion Network ==----------
http://www.dejanews.com/ Search, Read, Discuss, or Start Your Own

---

Subject: Re: Random numbers from a sound card?
Date: Tue, 26 Jan 1999 22:13:11 GMT
From: ross@hypertools.com (David Ross)
Message-ID: <36ae33b7.16083306@news.willapabay.org>
References: <78l7je$cc6$1@nnrp1.dejanews.com>
Newsgroups: sci.crypt
Lines: 60

Randombit -

  Thanks for an informative post.

On Tue, 26 Jan 1999 20:12:38 GMT, randombit@my-dejanews.com wrote:
> ross@hypertools.com (David Ross) wrote:
>> Has anyone had success using a sound card (like a Sound Blaster) to
>> generate streams of random numbers?
...
>> What sort of audio source would you suspect would be the best to use
>> in generating random numbers?
>
>I used an old radio shack mono fm radio, with antenna removed,
>tuned to hiss, at high volume, fed into the sound card.
>
>Later I got a video/radio digitizer, which I can tune to
>FM hiss, which is more self contained.
>
>This produces an apparently uniformly distributed noise spectrum,
>using a pc-based spectrum analyzer.
>
>But this doesn't have full entropy, and you have to distill (see
>RFC 1750) the bits. I experimented with parity-of-N bits,

>and used Maurer's Universal statistical test for RNGs to measure
>the entropy. When you distill enough, the entropy reaches its
>expected value.
>
>Some people might recommend a strong hashing function (e.g., a thousand
>raw bits hashed with MD5 down to a fixed output size). This is
>complex and I found unnecessary; simple parity works, though it
>may waste more bits than a serious hash would. But bits are cheap,
>and xor is fast.
  Lets say I'm digitizing a sine wave of constant frequency &
amplitude. If it has a 1 Volt peak amplitude and if I digitize at a
constant rate, won't 1/2 of my samples yield a value of either >+.707
Volt or < -.707 Volt? Seems like a built-in bias toward higher
numbers...
  After looking at 'random' noise on an oscilloscope, I'd expect to
see this same bias when digitizing noise...
  Digitizing either a sawtooth or a triangle waveform should get rid
of this inbuilt bias, but where to find 'sawtooth noise' is beyond me.

>> How would you test the 'quality' of the generated random number
>> stream?
>1. Marsaglia's Diehard suite of statistical (structure) tests.
>2. Maurer's Universal statistical test
>As "calibration standards" I used the "RAND million normal digits and their
>deviant friends", and also block ciphers run in feedback modes (ie,
>as PRNGs).
  Thanks for these suggestions.

>Also note that a 'loud' source of hiss is preferable.
  A 'loud' source of hiss may put the range of the A->D converter down
lower on a sinusoidal waveform. In this more linear area of the
waveform, you would get a more even distribution of digitized values
and begin to eliminate some of the inbuilt bias mentioned above.

David Ross ross@hypertools.com

Subject: Re: Random numbers from a sound card?
Date: Wed, 27 Jan 1999 18:09:54 GMT
From: randombit@my-dejanews.com
Message-ID: <78nkp9bol1@nnrp1.dejanews.com>
References: <36ae33b7.16083306@news.willapabay.org>
Newsgroups: sci.crypt
Lines: 28

In article <36ae33b7.16083306@news.willapabay.org>,
  ross@hypertools.com (David Ross) wrote:

> Lets say I'm digitizing a sine wave of constant frequency &
> amplitude. If it has a 1 Volt peak amplitude and if I digitize at a
> constant rate, won't 1/2 of my samples yield a value of either >+.707
> Volt or < -.707 Volt? Seems like a built-in bias toward higher
> numbers...

The threshold of your detector, which decides whether a voltage
is to be called a 0 or a 1 (for this clock period), will determine your
0:1 bias. (In addition to your raw waveform's properties!)

This threshold will interact with DC biases in your waveform.
And both will drift, and differ between parts.

You will not ever get perfect 0:1 ratios
---for instance, amplifiers may switch from 0->1 faster than 1->0
(even for CMOS) so you must plan for it. Happily, combining
multiple bits (e.g., parity) brings the ratio near
1:1 exponentially fast. Again, RFC 1750 is the bible.
Shannon is the Prophet.

----------== Posted via Deja News, The Discussion Network ==----------
http://www.dejanews.com/ Search, Read, Discuss, or Start Your Own

---

Subject: Re: Random numbers from a sound card?
Date: Thu, 28 Jan 1999 18:10:41 GMT
From: randombit@my-dejanews.com
Message-ID: <78q96o$ka6$1@nnrp1.dejanews.com>
References: <36AF1E85.280B09D4@stud.uni-muenchen.de>
  <78l7je$cc6$1@nnrp1.dejanews.com>
Newsgroups: sci.crypt
Lines: 32

In article <36AF1E85.280B09D4@stud.uni-muenchen.de>,
  Mok-Kong Shen <mok-kong.shen@stud.uni-muenchen.de> wrote:
> randombit@my-dejanews.com wrote:
> >
>
> > Note that even using a highly structured signal (e.g., digitized
> > video program including your local receiver noise) you could generate
> > good bits, but you'd have to distill bushels of them.

>
> I find your experience interesting. (In another thread I suggested
> obtaining good bit sequences from such materials as natural
> language texts.)
>
> M. K. Shen
>

There is a big difference. Eve does not know the local, instantaneious
electromagnetic conditions around my receiver, nor does she know what
my local electronics are doing.

The point is that measuring something physical is nothing like
playing with text streams, unless you you get them via UDP and
have a real bad link :-)

"Properly done science is a sort of masochistic game where one beats
one's head against a wall until it falls down, and then goes in search
of another wall." --Steven Vogel

----------== Posted via Deja News, The Discussion Network ==----------
http://www.dejanews.com/ Search, Read, Discuss, or Start Your Own

---

Subject: Re: Random numbers from a sound card?
Date: Thu, 28 Jan 1999 19:54:22 +0100
From: Mok-Kong Shen <mok-kong.shen@stud.uni-muenchen.de>
Message-ID: <36B0B25E.9FD6F2A@stud.uni-muenchen.de>
References: <78q96o$ka6$1@nnrp1.dejanews.com>
Newsgroups: sci.crypt
Lines: 22

randombit@my-dejanews.com wrote:
>

> There is a big difference. Eve does not know the local, instantaneious
> electromagnetic conditions around my receiver, nor does she know what
> my local electronics are doing.
>
> The point is that measuring something physical is nothing like
> playing with text streams, unless you you get them via UDP and
> have a real bad link :-)

In principle you are right. However, I assume that the choice of the
(publically known) texts that go into the process of generating
the bit sequences is secret information and can't be guessed by the
analyst. Thus, assuming adequate processing, one should obtain things
appropriate for use. Of course, one does not generate in this
way the legendary ideal OTP. But security is in my humble opinion
an issue dependent also on the cost and the like and I believe
that under certain real circumstances obtaining bit sequences from
software can be justified.

M. K. Shen

Subject: Re: Random numbers from a sound card?
Date: Thu, 28 Jan 1999 21:36:44 GMT
From: "patix" <patix@friko.onet.pl>
Message-ID: <ML4s2.5136$014.554762@news.tpnet.pl>
References: <78q96o$ka6$1@nnrp1.dejanews.com>
Newsgroups: sci.crypt
Lines: 27

randombit@my-dejanews.com wrote in message
<78q96o$ka6$1@nnrp1.dejanews.com>...
>In article <36AF1E85.280B09D4@stud.uni-muenchen.de>,

>
>There is a big difference. Eve does not know the local, instantaneious
>electromagnetic conditions around my receiver, nor does she know what
>my local electronics are doing.
>
>The point is that measuring something physical is nothing like
>playing with text streams, unless you you get them via UDP and
>have a real bad link :-)

OK , but if she assume that it is 50Hz (Europe power) and
let 30 KHz from Your monitor display , and if it happen to be true ?

I have question:Haw should we test hardawre random
generator to "be shoure" that it is realy some haw random ?

patix

Subject: Re: Random numbers from a sound card?
Date: 29 Jan 1999 08:44:58 -0500
From: juola@mathcs.duq.edu (Patrick Juola)
Message-ID: <78se0q$or1$1@quine.mathcs.duq.edu>
References: <78q96o$ka6$1@nnrp1.dejanews.com>
Newsgroups: sci.crypt
Lines: 25

In article <78q96o$ka6$1@nnrp1.dejanews.com>,
 <randombit@my-dejanews.com> wrote:
>In article <36AF1E85.280B09D4@stud.uni-muenchen.de>,
> Mok-Kong Shen <mok-kong.shen@stud.uni-muenchen.de> wrote:
>> randombit@my-dejanews.com wrote:
>> >
>>
>> > Note that even using a highly structured signal (e.g., digitized
>> > video program including your local receiver noise) you could generate
>> > good bits, but you'd have to distill bushels of them.
>>
>> I find your experience interesting. (In another thread I suggested
>> obtaining good bit sequences from such materials as natural
>> language texts.)
>>
>
>There is a big difference. Eve does not know the local, instantaneious
>electromagnetic conditions around my receiver, nor does she know what
>my local electronics are doing.

No, but Mike does. He's fully capable of broadcasting (known)
noise of some sort near your site.

        -kitten

Subject: Re: Random numbers from a sound card?
Date: Tue, 26 Jan 1999 22:55:51 +0000
From: pla@sktb.demon.co.uk (Paul L. Allen)
Message-ID: <f8pGhDj030n@sktb.demon.co.uk>
References: <36AE059A.9BA@physiology.wisc.edu>
   <36acb8b1.5374650@news.willapabay.org>
Newsgroups: sci.crypt
Lines: 13

In article <36AE059A.9BA@physiology.wisc.edu>
    Medical Electronics Lab <rosing@physiology.wisc.edu> writes:

> DIEHARD from Marsaglia

I can never find a URL for that when I need it. I saw Marsglia posted
his various PRNGs the other week and mentioned it but no URL. Actually,
I am more interested in the accompanying docs than the actual tests right
now. What a shame we can't get the FAQ updated.

--Paul

Terry Ritter, *his current address*, *and his top page*.

Last updated: 1999-02-20

The Several Types of Random

A *Ciphers By Ritter* Page

A discussion of the term "truly random."

Contents

- 1998-11-06 Horst Ossifrage: "Consider the difference between 'random' and 'truely random'. There is no difference...."
- 1998-11-07 W T Shaw: "I find your definitions 'slanted.'"
- 1998-11-10 Scott Nelson: "I disagree, there is a difference. Just as there's a difference between 'cat' and 'house cat'...."
- 1998-11-10 newWebsite: "1) Are you certain of the definitions? 2) Are you truely certain of the definitions?"
- 1998-11-11 Scott Nelson: "'true random' has a specific idiomatic meaning, just as 'true colors' means something different from 'colors.'"
- 1998-11-12 Mike McCarty: "Unfortunately, the word 'random' by itself has no particular meaning."

```
Subject: random: Glossary Term
Date: Fri, 06 Nov 1998 22:06:25 -1000
From: Horst Ossifrage &LT;WWII@wwww.com>
Message-ID: &LT;3643FF81.593E@wwww.com>
Newsgroups: sci.crypt
Lines: 92

random: Glossary Term 3 of 5
____________________________

The word "random" is easily understood by most readers of the sci.crypt
newsgroup. I will not define it here, but a few comments need to be
published, anyway. Consider the difference between "random" and
"truely random". There is no difference, and yet people write
scholarly papers, and are not satisfied to call numbers
random, they need to emphasize by calling a sequence
of numbers "truely random". That practice should be
stopped: by editors, or by style convention sheets
that are commonly distributed to authors. There
are three main classes of numbers considered
here: random numbers, pseudo-random numbers,
and non-random numbers. (See Terry Ritter's
Glossary for more discussions). One more
point: some sequences "seem random to
people" and some "seem non-random".
But that perception has almost no
value in the realm of keys and
ciphers. Flipping a coin has
a human influence and seems
to make random numbers but
other forms of human
influence seem to
be non-random.

The "Osprey Protocol"
_____________________
The Osprey Protocol is a way to distribute keys and ciphertexts
on insecure channels, like e-mail and newsgroups. Many varia-
tions are possible, but the one described here uses 2-level
super-encryption. A plaintext is encrypted with scott19u
using the keyraw.key file that is created by cutting and
pasting an entire message, like "mudge: Glossary Term
2 of 5" posted 11/5/1998, on sci.crypt, with the
"space counts" added. That ciphertext is then
encrypted using PGP 2.6.2 (IDEA) using a key
that is composed of the "space counts".
Space counts are defined by Rule 3,
below. The PGP outputs ASCII letters
as ciphertext by using the -a option. A
chaining strategy is recommended, where the
keys are in one message, without the corresponding
ciphertext, and then a second message has the next key,
and the ciphertext for the first key.

Rule 1: The paragraph shape can be
used to indicate to the initiates that
the Osprey Protocol is being used. For example,
a monotonically decreasing paragraph shape is the
rule this week. Next week's rule will be indicated
in later ciphertexts. This way, no explicit statements
need to be made to alert initiates to the existence of
the protocol.

Rule 2: Use only the first paragraph for space counts.

Rule 3: Determine the "space counts" as follows: add one
space after the longest line and put a 1 after the space.
The numbers are right adjusted as the spaces are counted for
each line, and the count is typed at the end of each line.
Try it for the first paragraph in this essay.

Rule 4: String together all of the space counts to make the
key for IDEA. If there are less than 16 digits, pad with zeros.
If there are more than 16 digits, use the first 16.

Discussion
__________
The "mudge: Glossary Term" posting has over 1200 characters as
the key file for scott19u. The key for the IDEA step has 128 bits
or 16 ASCII characters. Use the 16 digits of the "space counts"
as the key for IDEA. Here is the ciphertext that you may decrypt
using the Osprey Protocol:
-----BEGIN PGP MESSAGE-----
Version: 2.6.2

pgAAAHjOfUx2V7TUJlGWodjtSIrN0nmrxo3Xhe+LPHZTYeAedCk27mKonpljOIRO
zJszElBiZpzdf3eAOATHRC2WEpC09CstduEV0fkoP5bLvetDRofbKzpRbzwF9uTM
ohrPYZ5ZnFR00IdOXJfe7/1BqXBF82TEuyObzns=
=BSE6
-----END PGP MESSAGE-----
Use this current posting for the keys for tomorrows part 4 of 5
Glossary Term postings in sci.crypt. In scott19u, use the menu
choice with no padding, and when asked for a key, he means
give a password. Press ENTER without giving a password.
November 6, 1998 9:55PM
Horst Ossifrage
San Jose
USA
!
```

```
Subject: Re: random: Glossary Term
Date: Sat, 07 Nov 1998 10:57:44 -0600
From: jgfunj@EnqvbSerrGrknf.pbz (W T Shaw)
Message-ID: &LT;jgfunj-0711981057450001@dialup172.itexas.net>
References: &LT;3643FF81.593E@wwww.com>
Newsgroups: sci.crypt
Lines: 7


I find your definitions "slanted."
--
---
The public is harder to steamroller than some might think.
---
Decrypt with ROT13 to get correct email address.
```

```
Subject: Re: random: Glossary Term
Date: Tue, 10 Nov 1998 04:53:15 GMT
From: scott@helsbreth.org (Scott Nelson)
Message-ID: &LT;3648bf9a.92225357@news.inreach.com>
References: &LT;3643FF81.593E@wwww.com>
Newsgroups: sci.crypt
Lines: 42

On Fri, 06 Nov 1998, Horst Ossifrage &LT;WWII@wwww.com> wrote:
>
>The word "random" is easily understood by most readers of the sci.crypt
>newsgroup.
>
Unfortunately, it's also easily misunderstood by many readers.

>I will not define it here, but a few comments need to be
>published, anyway. Consider the difference between "random" and
>"truely random".
Ok.  The main difference I see is that random is a general
term, used to denote a large class of things including
some things (such as pseudo random number generators)
which don't have all of the properties needed for "true"
randomness.

>There is no difference, and yet people write
>scholarly papers, and are not satisfied to call numbers
>random, they need to emphasize by calling a sequence
>of numbers "truely random".
I disagree, there is a difference.  Just as there's
a difference between "cat" and "house cat" or
"number" and "integer."

>That practice should be
>stopped: by editors, or by style convention sheets
>that are commonly distributed to authors.
Again I disagree.  It's very useful to be able
to use short words for discussion, with the
exact form taken from context.  If by convention
"random" always meant "truly random" then talking
about pseudo random would be very cumbersome indeed.

And since I'm posting anyway, here a link to
Ritter's glossary, which is, IMO, excellent.
Thanks Terry for providing it.
http://www.io.com/~ritter/GLOSSARY.HTM

-----------------------------------
Shameless plug for random web site:
http://www.helsbreth.org/random
Scott Nelson &LT;scott@helsbreth.org>
```

```
Subject: Re: random: Glossary Term
Date: Tue, 10 Nov 1998 22:32:25 -1000
From: newWebsite &LT;polite@yahoo.com>
Message-ID: &LT;36494B99.27C@yahoo.com>
References: &LT;3648bf9a.92225357@news.inreach.com>
Newsgroups: sci.crypt
Lines: 43

Scott Nelson wrote:

> Ok.  The main difference I see is that random is a general
> term, used to denote a large class of things including
> some things (such as pseudo random number generators)
> which don't have all of the properties needed for "true"
> randomness.
>
> >There is no difference, and yet people write
> >scholarly papers, and are not satisfied to call numbers
> >random, they need to emphasize by calling a sequence
> >of numbers "truely random".

> I disagree, there is a difference.  Just as there's
> a difference between "cat" and "house cat" or
> "number" and "integer."
>
> >That practice should be
> >stopped: by editors, or by style convention sheets
> >that are commonly distributed to authors.

> Again I disagree.  It's very useful to be able
> to use short words for discussion, with the
> exact form taken from context.  If by convention
> "random" always meant "truly random" then talking
> about pseudo random would be very cumbersome indeed.

Please answer 2 questions:

1) Are you certain of the definitions?

2) Are you truely certain of the definitions?

If you can answer yes to one and no to the other, then you have a good
argument to convince me that there is a difference between random and
truely random. I am waiting...


--
See the newWebsite on A Large-Key Encryption algorithm's
inner workings! http://members.xoom.com/ecil/ for the
finest file scrambling in the known universe!
Expect nothing, get everything.
```

```
Subject: Re: random: Glossary Term
Date: Wed, 11 Nov 1998 16:59:08 GMT
From: scott@helsbreth.org (Scott Nelson)
Message-ID: &LT;364ab715.879846@news.inreach.com>
References: &LT;36494B99.27C@yahoo.com>
Newsgroups: sci.crypt
Lines: 96

On Tue, 10 Nov 1998 newWebsite &LT;polite@yahoo.com> wrote:

>Scott Nelson wrote:
>
>> Ok.  The main difference I see is that random is a general
>> term, used to denote a large class of things including
>> some things (such as pseudo random number generators)
>> which don't have all of the properties needed for "true"
>> randomness.
>>
>> >There is no difference, and yet people write
>> >scholarly papers, and are not satisfied to call numbers
>> >random, they need to emphasize by calling a sequence
>> >of numbers "truely random".
>
>> I disagree, there is a difference.  Just as there's
>> a difference between "cat" and "house cat" or
>> "number" and "integer."
>>
>> >That practice should be
>> >stopped: by editors, or by style convention sheets
>> >that are commonly distributed to authors.
>
>> Again I disagree.  It's very useful to be able
>> to use short words for discussion, with the
>> exact form taken from context.  If by convention
>> "random" always meant "truly random" then talking
>> about pseudo random would be very cumbersome indeed.
>
>Please answer 2 questions:
>
>1) Are you certain of the definitions?
>
>2) Are you truely certain of the definitions?
>
>If you can answer yes to one and no to the other, then you have a good
>argument to convince me that there is a difference between random and
>truely random. I am waiting...

Gee, I'd hate to keep you waiting . . .
1.) Yes.
2.) No.
(If you want, I'll even answer yes and no to the same question. :-)

Normally, I just gloss over people who want to argue semantics,
but in this case, I think semantics is the whole point.
I'll expand on my original statements.

  In my definitions of random,
( http://www.helsbreth.org/random/ )
I make a distinction between "true random" "cryptographically
strong random" and "pseudo random"  These terms come from a
scanning of the literature.  Now if I were choosing, I
wouldn't choose "true" to describe the unreproducible
property, but it's not my choice to make.  The damage has
already been done.

  There's some confusion between the common usage of
"true" with the usage normally associated with random.
In other words, "true random" has a specific
idiomatic meaning, just as "true colors" means
something different from "colors."  Because english
is extensible, truly random might have the
more limited meaning of "a random generator having
the property of being able to produce different outputs
given the same starting conditions."   Random might
refer to a pseudo random generator, truly random can not.


  Now one could argue that random shouldn't include
pseudo random, or cryptographically strong random.
Random should be used only for things which
measure up to the ideal of randomness; unbiased,
unpredictable, and unreproducible.  While I agree
that it's an arguable position, it happens to be
one I disagree with.  I prefer to keep discussions
brief and to the point.  Short words and phrases
should be assigned to those concepts we use most
often.  We don't talk about the ideal randomness
very often, while pseudo random generators are
a dime a dozen.  If we were to affix a single
meaning to random, I would think ideal randomness
would be a poor choice.

  Better still is to allow random to a be dynamically
assigned within the context of the discussion at hand.
This is typically done by using the precise term at
the beginning of the paper, then using the short form
in the remainder.  I like it that way, which is
why I took the time to argue against the original
poster in the first place.

-----------------------------------
Shameless plug for random web site:
http://www.helsbreth.org/random
Scott Nelson &LT;scott@helsbreth.org>
```

```
Subject: Re: random: Glossary Term
Date: 12 Nov 1998 16:10:23 GMT
From: jmccarty@sun1307.spd.dsccc.com (Mike McCarty)
Message-ID: &LT;72f19f$72s$1@relay1.dsccc.com>
References: &LT;3643FF81.593E@wwww.com>
Newsgroups: sci.crypt
Lines: 46

In article &LT;3643FF81.593E@wwww.com>, Horst Ossifrage  &LT;WWII@wwww.com> wrote:
)random: Glossary Term 3 of 5
)____________________________
)
)The word "random" is easily understood by most readers of the sci.crypt
)newsgroup. I will not define it here, but a few comments need to be
)published, anyway. Consider the difference between "random" and
)"truely random". There is no difference, and yet people write
)scholarly papers, and are not satisfied to call numbers
)random, they need to emphasize by calling a sequence
)of numbers "truely random". That practice should be
)stopped: by editors, or by style convention sheets

[snip]

Unfortunately, the word "random" by itself has no particular meaning. It
is not usefully used as an independent adjective. It has meaning only
when used in combination with other words. For example these terms all
have well-defined meanings:

        random variable
        random sample
        random sequence
        random (or stochastic) process

None of these terms have any well-defined meaning

        random number
        random person
        random house (sorry, couldn't resist)

In short, "random" is not really a word in itself. (I'm talking in
pedagogical terms, I don't mean that it can't/shouldn't be used in
casual speach.)

Also unfortunately, as typified by random variable, the special
meanings the special terms have are not particularly associated with
being random. For example, a random variable is neither random, nor is
it a variable; it is a function.

Mike
--
----
char *p="char *p=%c%s%c;main(){printf(p,34,p,34);}";main(){printf(p,34,p,34);}
This message made from 100% recycled bits.
I don't speak for Alcatel      &LT;- They make me say that.
```

Terry Ritter, his *current address*, and his *top page*.

Last updated: 1999-01-19

Junction Noise Experiments

A *Ciphers By Ritter* Page

Terry Ritter

Subject: Noise Experiments
Date: 21 Aug 1994 23:56:03 -0500
From: ritter@io.com (Terry Ritter)
Message-ID: <199408220456.XAA28923@pentagon.io.com>
Newsgroups: sci.crypt
Lines: 226

NOISE EXPERIMENTS

I recently spent the better part of several nights and days trying to see just how easy it really is to generate noise from a reverse-biased transistor B-E junction. I breadboarded several different circuits, and measured the amount of noise produced.

The method does seem practical, but uses a number of parts and may require some testing and/or adjustment, because the noise depends upon non-guaranteed transistor characteristics.

Day One

My first successful circuit was:

```
            _ V+                        _
            |                           |
            |                           R3
            R1                          |
            |                           +---> V2
         V1 |                           c
            |--------+--R2--+-----b Q2
            e        |     |     e
         +--b Q1     +--C1--+     |
         |  c                      v
         |  |
         v  v
```

Both transistors are NPN silicon.

Q1 pretends to be a zener, and drops current across R1, presumably creating noise in the process. R2 biases Q2, but is so large that C1 is needed as a bypass. R3 is probably not critical.

A production design would probably bias Q2 with negative feedback and just couple the noise in with C1.

Here are the results for two configurations:

	V+	Q1	R1	V1	R2	C1	Q2	R3	V2	AC
1/	11.3v	2N5088	100K	6.65v	1M	.22	2N2222	12K	5.0v	56mv
2/	10.0v	2N2222	100K	7.04v	1M	.22	2N2222	12K	3.2v	168mv

AC is the RMS noise output. Note that RMS is the power in the noise signal, and not the peak-to-peak voltage, which may be more related to what we need for subsequent processing. 56mv RMS seems to show an "average" amplitude of about 300mv p-p on the scope. Technically, the absolute amplitude exceeds the rms value about 1/3 of the time, with 5 rms exceeded about 1 time in a million, but the scope probably has a much larger bandwidth than the meter.

If we work out the currents in 1/, Q2 must be running with a beta around 88, and if we assume that the small-signal gain is the same, we have about 640uv of RMS noise on Q1. Similarly, /2 shows Q2 with a beta around 89, and 1.88mv of RMS noise on Q1.

Is AC really noise? Removing C1, or connecting it across Q1 had no affect on the DC levels, but reduced the AC value to the background level of 4mv. (The RMS meter showed 4mv on the supply.) Thus, the circuit was apparently neither oscillating nor collecting a significant amount of RF.

In /2, R1 was changed to 10K and 470K; in both cases the noise was drastically reduced. Apparently there is some optimum current for maximum noise.

Also in /2, Q1 was replaced with several different zener diodes. In all cases, the AC noise output dropped to background. Changing R1 to 10K did not seem to help. It is tempting to think that a transistor in "zener connection" is exceptionally noisy, but maybe real zeners just need more power.

Day Two

A noise source with a self-biasing amp:

```
            _ V+                        _
            |                           |
            |                           R2
            R1                          |
            |         +--R4--+--R3--+---> V2
            |         |      |       |
            |         |      C2      |
            |         |      v       |
         V1 |         |              c
            |----C1---+------------b Q2
            e                        e
         +--b Q1                     |
         |  c                        v
         |  |
         v  v
```

Both transistors 2N2222

V+	R1	V1	C1	R2	R3	C2	R4	V2	AC
10.0v	100K	7.04v	.22uf	12K	100K	10uf	100K	2.1v	168mv

This gives a very reasonable output, at the expense of no automatic biasing for Q1. On the other hand, the biasing may be fairly stable, so this could be a good configuration.

We can also auto-bias the noise source. This circuit is very similar to one in an article called "Truly Random Numbers" by H.R. Bungay and Robert Martin in an early issue of Kilobaud:

```
                            _ V+
                            |
                            R1
            V1              |
      +------+------R3-----+ V2
      |      |              |
      C1     |              R2
      |      |              |
      v      |              +---> V3
             |              |
             e              c
         Q1 b------------b Q2
             c              e
             (nc)           |
                            v
```

Both transistors are 2N2222.

V+	R1	R2	R3	C1	V1	V2	V3	AC
10.0v	12K	15K	68K	10uf	7.53v	7.7v	4.75v	80mv

Note that this circuit trades off output amplitude for automatic biasing.

The fixed zenering level fixes V1; feedback fixes V2 based on V1; R2 serves to move the output voltage nearer to center.

This particular E-B junction appears to break down about 7v, but this is not a guaranteed quantity. The 2N2222 has a guaranteed minimum Vbe of 5v (a common value); I searched around and found a 2N1990 with a minimum guaranteed Vbe of 3v, but this particular transistor happened to have an even "higher" breakdown. It would be nice to find a "transistor" with a guaranteed low breakdown (say 2.5v) and then try to work from a logic level supply, but with the present components, a low ripple 9v or higher voltage supply seems necessary.

If 80mv (RMS!) is not enough, we can try a simple amplifier:

```
                            _ V+
                            |
                            R4
                            |
                     +--R5--+-+--> V4
                     |      |
                     |      c
   ---C2-----R6---+------b Q3
                     |+     e
                     D1     |
                     |      v
                     v
```

Q3 is 2N2222; D1 is a switching diode.

V+	R4	R5	C2	R6	V4	AC
10.0v	4.7K	220K	.22uf	0K	4.17v	1.07v

Now we get peaks virtually to the 0 and 10v extremes. The scope seems to show a bandwidth change, however, so we might need to reduce C2 and increase C6, remove D1, or perhaps even use a high-frequency transistor.

R5 tends to steady the output voltage to a reasonable mid value.

Alternatives

One alternative would be to use an op-amp to perform the final amplification; possibly it could perform limiting and digital conversion as well.

Of course, if we are going to support one op-amp, we might as well support others, and then can eliminate all but the noise-source transistor. Of course the op-amp circuits will have their own costs.

The number of components and the variability of the noise source in the reverse-biased junction approach may make the FM IF strip idea a very interesting alternative.

Conclusion

Several simple circuits can generate really-random noise from a reverse-biased transistor B-E junction, and then amplify this noise to near usable levels. Subsequent limiting and then pulse-width measurement (for example) would be one way to use this noise to generate really random values.

Terry Ritter ritter@io.com

Terry Ritter, his *current address*, and his *top page*.

Last updated: 1999-01-16

Nico's Really Random Generator

A massive conversation starting with a proposal for generating "really random" numbers in the PC environment without added hardware.

- 1992-06-19 Nico de Vries: A "flawless, noise-based" really-random generator for the PC using software only?
- 1992-06-20 Nico de Vries: Nico responds to Tony Patti's message about his RANGER really-random hardware using 16 crystal oscillators (the earlier message was not saved)
- 1992-06-20 Tony Patti: Tony responds to Nico on the RANGER design
- 1992-06-22 Nico de Vries: (get the "trick" by e-mail)
- 1992-06-22 Paul Koning: Paul responds to Nico about crystal oscillator correlation
- 1992-06-22 Nico de Vries: Nico responds to Paul: "try it"
- 1992-06-22 Nico de Vries: Nico finally publishes the source
- 1992-06-23 Terry Ritter: Terry jumps in with an early analysis
- 1992-06-23 Nico de Vries: Nico responds "I couldn't find any patterns"
- 1992-06-23 Vesselin Bontchev: Vesselin suggests using a visual display for detecting adjacent-value correlation
- 1992-06-23 Nico de Vries: Nico's visual correlation test source
- 1992-06-23 Terry Ritter: Terry suggests that memory refresh DMA "could be the major source of apparent randomness"
- 1992-06-24 Phil Karn: "Surely Nico's algorithm is better than many existing techniques"
- 1992-06-24 Nico de Vries: "Can anyone with a single clock please test the code"
- 1992-06-24 Terry Ritter: "crystal oscillators do not 'jitter'"
- 1992-06-24 Ross Anderson: "The current flame war"
- 1992-06-24 Nico de Vries: "no flame war"
- 1992-06-25 Terry Ritter: "no flame war"
- 1992-06-25 Ted Drawneek: "phase noise is present even in crystal oscillators"
- 1992-06-25 Nico de Vries: "is a crystal 'fully deterministic'"
- 1992-06-25 Nico de Vries: Nico responds to Ted: (phase noise" is "what I expected"
- 1992-06-25 Nico de Vries: Nico responds to Terry: "agree no flame war"
- 1992-06-25 Terry Ritter: Terry responds to Ted: "phase noise is 2.5 psec. Measure *that* on a PC with software and timers"
- 1992-06-25 Terry Ritter: "for practical purposes, using PC timers and software measurement, crystal oscillators do not 'jitter'"
- 1992-06-26 Ted Drawneek: Ted responds to Terry: "phase noise can have a bandwidth of 3 kHz"
- 1992-06-26 Phil Zimmermann: "let's use MD5 to distill down true randomness"
- 1992-06-26 Colin Plumb: Colin displays his knowledge of RNG's
- 1992-06-26 Nico de Vries: Nico responds to Terry: "no practical test has shown problems"
- 1992-06-26 Terry Ritter: Terry responds to Colin
- 1992-06-26 Terry Ritter: Terry responds to Phil Z.
- 1992-06-27 Phil Zimmermann: Phil Z. responds to Terry
- 1992-06-27 Johnathan Thornburg: Johnathan weighs in
- 1992-06-27 Terry Ritter: Terry finally bestirs himself to present a serious analysis
- 1992-06-27 Phil Karn: "widen the counter and run an autocorrelation on the results"
- 1992-06-28 Albert Boulanger: Albert weighs in
- 1992-06-28 Greg Rose: Greg supports Nico
- 1992-06-29 Miroslav Asic: apparently Miroslav's hot button is CRC "strength"
- 1992-06-29 Nico de Vries: "the generator is useful"
- 1992-06-29 Nico de Vries: P.S.
- 1992-07-01 Peter Gutmann: Peter suggests that there are other PC oscillators which could also be used
- 1992-07-01 Ralph Neutrino: "chaotic dynamical systems are definitely not random"
- 1992-10-26 Tony Patti: Tony summarizes the features of his RANGER really-random design
- 1992-11-03 Terry Ritter: constitutionally unable to let it lie, Terry presents Crystal Oscillators Thought Harmful

---

Terry Ritter, his current address, and his top page.

Last updated: 1995-10-29

Birthday Attack Calculations

A *Ciphers By Ritter* Page

How can we relate the number of elements in a population and the number of random samples needed before we expect to find a duplicate or match?

Contents

- 1998-12-30 Fred Van Andel: "How does one calculate the exact number of documents that need to be hashed to ensure a collision?"
- 1998-12-30 DJohn37050: "...the exact number is n+1."
- 1998-12-31 Fred Van Andel: "I should have stated for 50% probability...."
- 1998-12-03 Adam Atkinson: "...doing it "by hand" in perl would look something like...."
- 1999-01-2 David Broughton: "Try this formula...."
- 1999-01-2 Peter Pearson: "To derive your own approximation formula...."
- 1999-01-02 Terry Ritter: "The formula basically comes out of my article on population estimation...."
- 1999-01-06 Fred Van Andel: "...I am designing a variable length hash function and I want to test it for resistance to collision."
- 1999-01-06 dscott@networkusa.net: "This is a very good idea. Since only if hash is any good will it have the distribution predicticed by the bithday collision method."
- 1999-01-07 Fred Van Andel: "The whole point of testing on small hashes and extrapolating is that is computationally impossible to do a birthday attack on a large hash."
- 1999-01-07 dscott@networkusa.net: "...it would be stupid not to run some tests in hopes that no collision occur at long length."
- 1999-01-09 dscott@networkusa.net: "...the thing was worded that he should never test the FULL LENGTH case...."
- 1999-01-09 Fred Van Andel: "If the hash is so fatally flawed that I could routinely find a collision at much lower than the normal values then I would have found this out at the smaller hash sizes."
- 1999-01-09 dscott@networkusa.net: "Then run 100 cases after your done with your low level testing."
- 1999-01-10 dscott@networkusa.net: "...you do need to try a few cases just to verify that ZERO occur...."
- 1999-01-09 Fred Van Andel: "The smaller hash sizes are not from a 'cut-down' algorithm. The nature of this algorithm is that is can create hashes of any desired size...."
- 1999-01-09 Trevor Jackson, III: "Either the attack is not a birthday attack, or the odds of a collision are much higher."
- 1999-01-10 Trevor Jackson, III: "...I assumed that each sample would be stored after comparison with the previously stored samples."
- 1999-01-11 Fred Van Andel: "A birthday attack would require > 2^128 calculations while an exhaustive search would need 2^255."
- 1999-01-11 Trevor Jackson, III: "Is there a closed-form equation for the figure 2^128 you quoted?"
- 1999-01-12 Fred Van Andel: "For any gives value of M the location of the 50% mark will be roughly the square root of M."
- 1999-01-12 Trevor Jackson, III: "...I am slightly confused over the odds of finding the first collision and the odds of finding no or any collisions."
- 1999-01-13 Fred Van Andel: "For a value of 1,000,000, i is 1178."
- 1999-01-13 Terry Ritter: "Which seems to compare rather well to the value 1177.9 computed from my formula in the earlier posting...."
- 1999-01-14 Fred Van Andel: "Why do you think I chose the number 1,000,000."

```
Subject: Birthday Attack calculations.
Date: Wed, 30 Dec 1998 23:16:02 GMT
From: fred_vanandel@bigfoot.com (Fred Van Andel)
Message-ID: &LT;368ab32b.57279906@news.intergate.bc.ca>
Newsgroups: sci.crypt
Lines: 7

How does one calculate the exact number of documents that need to be
hashed to ensure a collision?  I know that it is approximately
1.2*2^(M/2), but what is the exact formula or procedure for
calculating the number?

Thanks
Fred Van Andel
```

```
Subject: Re: Birthday Attack calculations.
Date: 30 Dec 1998 23:59:16 GMT
From: djohn37050@aol.com (DJohn37050)
Message-ID: &LT;19981230185916.28531.00004724@ng39.aol.com>
References: &LT;368ab32b.57279906@news.intergate.bc.ca>
Newsgroups: sci.crypt
Lines: 3

To ensure a collision, the exact number is n+1.  THe birthday attack in
probabilistic.  See the HAC.
```

```
Subject: Re: Birthday Attack calculations.
Date: Thu, 31 Dec 1998 03:46:01 GMT
From: fred_vanandel@bigfoot.com (Fred Van Andel)
Message-ID: &LT;368af32e.73668929@news.intergate.bc.ca>
References: &LT;19981230185916.28531.00004724@ng39.aol.com>
Newsgroups: sci.crypt
Lines: 10

djohn37050@aol.com (DJohn37050) wrote:

>To ensure a collision, the exact number is n+1.  THe birthday attack in
>probabilistic.  See the HAC.

I should have stated for 50% probability, the formula given is for
that value.  My aplogies.

Fred Van Andel
```

```
Subject: Re: Birthday Attack calculations.
Date: 31 Dec 98 04:55:33 +0000
From: "Adam Atkinson" &LT;ghira@mistral.co.uk>
Message-ID: &LT;563.669T1659T2953896@mistral.co.uk>
References: &LT;368af32e.73668929@news.intergate.bc.ca>
Newsgroups: sci.crypt
Lines: 28

On 31-Dec-98 03:46:01, Fred Van Andel said:

>>To ensure a collision, the exact number is n+1.  THe birthday attack in
>>probabilistic.  See the HAC.

>I should have stated for 50% probability, the formula given is for
>that value.  My aplogies.

well, doing it "by hand" in perl would look something like:

$n=43949268;
#$n=365; #gives 23, which is right

$p=1;

for($i=1;$p>0.5;$i++) {
$p*=($n-$i)/$n;
}
print $i."\n";

--
Adam Atkinson (ghira@mistral.co.uk)
Verbing weirds language. (Calvin)
```

```
Subject: Re: Birthday Attack calculations.
Date: Sat, 2 Jan 1999 14:41:56 +0000
From: David Broughton &LT;David@ddina.demon.co.uk>
Message-ID: &LT;$7zENAA0Ajj2EwNz@ddina.demon.co.uk>
References: &LT;368ab32b.57279906@news.intergate.bc.ca>
Newsgroups: sci.crypt
Lines: 36

In article &LT;368ab32b.57279906@news.intergate.bc.ca>, Fred Van Andel
&LT;fred_vanandel@bigfoot.com> writes
> How does one calculate the exact number of documents that need to
> be hashed to ensure a collision?  I know that it is approximately
> 1.2*2^(M/2), but what is the exact formula or procedure for
> calculating the number?

Try this formula:

    w = n^g + 0.29 - e

where:
w = the number of documents to get 50% probability of a collision
n = number of different hash codes, all equiprobable
g = 0.5 + 1/(6.13 * ln(n))
ln() is the natural logarithm function
e is a small error that can be ignored in practice, usually &LT; +- 1.

This is an empirical formula that I worked out many years ago and
filed away in a notebook.

The exact formula is the value of w in this equation:

  ( product from k = 1 to k = w-1 of (n-k)/n ) = 0.5

but this is not a practical calculation for large n.

As you can see, w is a bit larger than the square root of n.  For
n = 10^6, for example, w = 1177.68 (e = -0.197).

If your formula is meant to be 1.2 * n^0.5, then w = 1200 for n =
10^6 which is getting there.

--
David Broughton
```

```
Subject: Re: Birthday Attack calculations.
Date: Sat, 2 Jan 1999 19:11:25 GMT
From: pearson@netcom.com (Peter Pearson)
Message-ID: &LT;pearsonF4y5B1.Jyv@netcom.com>
References: &LT;$7zENAA0Ajj2EwNz@ddina.demon.co.uk>
Newsgroups: sci.crypt
Lines: 38

In article &LT;$7zENAA0Ajj2EwNz@ddina.demon.co.uk>,
David Broughton  &LT;David@ddina.demon.co.uk> wrote:
>In article &LT;368ab32b.57279906@news.intergate.bc.ca>, Fred Van Andel
>&LT;fred_vanandel@bigfoot.com> writes
>
>> How does one calculate the exact number of documents that need to
>> be hashed to ensure a collision?  I know that it is approximately
>> 1.2*2^(M/2), but what is the exact formula or procedure for
>> calculating the number?
>
> [snip]
>
>The exact formula is the value of w in this equation:
>
>   ( product from k = 1 to k = w-1 of (n-k)/n ) = 0.5
>
>but this is not a practical calculation for large n.

To derive your own approximation formula from the above, exact
formula, rewrite it as follows:

            n (n-1) (n-2) ... (n+1-w)          n!
  product = ------------------------- = ----------
                       n^w               (n-w)! n^w

Then, use Stirling's approximation to make the factorials more
manageable. Stirling's approximation (see, for example, Knuth,
Art of Computer Programming, Volume 1) is:

  ln( n! ) = (n+1/2) ln(n) - n + ln( 2*pi )/2 + 1/(12*n) - ...

You'll have to experiment with the number of terms required to
get meaningful results. Overimplifying to n^ln(n)-n gives conspicuously
nonsensical results. If memory serves, (n+1/2) ln(n) - n + ln( 2*pi )/2
is a good level of approximation, and one needs the approximation
ln(1+x) = x (for small x) as well.

- Peter
```

```
Subject: Re: Birthday Attack calculations.
Date: Sat, 02 Jan 1999 22:46:05 GMT
From: ritter@io.com (Terry Ritter)
Message-ID: &LT;368ea1e9.8344667@news.io.com>
References: &LT;368ab32b.57279906@news.intergate.bc.ca>
Newsgroups: sci.crypt
Lines: 40

On Wed, 30 Dec 1998 23:16:02 GMT, in
&LT;368ab32b.57279906@news.intergate.bc.ca>, in sci.crypt
fred_vanandel@bigfoot.com (Fred Van Andel) wrote:

>How does one calculate the exact number of documents that need to be
>hashed to ensure a collision?  I know that it is approximately
>1.2*2^(M/2), but what is the exact formula or procedure for
>calculating the number?

>[I should have stated for 50% probability, the formula given is for
>that value.]

I like:

   s(N,p) = (1 + SQRT(1 - 8N ln(p))) / 2

where s is the expected number of samples needed, N the size of the
population being sampled, and p the given probability.

For N = 10**6 and p = 0.5 (so ln(p) = -0.693) we get 1177.9 instead of
the usual handwave SQRT(N) = 1000, or the stated approximation, 1.2 *
SQRT(N) = 1200.

For N = 2**20 and p = 0.5, we get 1206.2 instead of 1.2 * 1024 =
1228.8 for the stated approximation.

The formula basically comes out of my article on population
estimation:

   http://www.io.com/~ritter/ARTS/BIRTHDAY.HTM

It might be interesting to know of an application where this sort of
precision is useful.

---
Terry Ritter   ritter@io.com   http://www.io.com/~ritter/
Crypto Glossary   http://www.io.com/~ritter/GLOSSARY.HTM
```

```
Subject: Re: Birthday Attack calculations.
Date: Wed, 06 Jan 1999 06:01:50 GMT
From: fred_vanandel@bigfoot.com (Fred Van Andel)
Message-ID: &LT;3693fa55.17551686@news.intergate.bc.ca>
References: &LT;368ea1e9.8344667@news.io.com>
Newsgroups: sci.crypt
Lines: 17

I would like to thank those who spent time in answering my question.
It has proven to be very helpful.

 Terry Ritter wrote:
>It might be interesting to know of an application where this sort of
>precision is useful.

The reason for the needed precision is that I am designing a  variable
length hash function and I want to test it for resistance to
collision.  Since I don't have the resources to do a full test  for a
256 bit hash I am going to test 8, 16, 24, 32 and maybe 40 bit hashes
and search for trends. If the algorithm is resistant to collision in
the smaller sizes and there is no trend away from the "proper" value
then due to the nature of the algorithm I can be quite confidant that
the larger hashes are also resistant to collisions.

Fred Van Andel
```

```
Subject: Re: Birthday Attack calculations.
Date: Wed, 06 Jan 1999 12:54:24 GMT
From: dscott@networkusa.net
Message-ID: &LT;76vme0$avi$1@nnrp1.dejanews.com>
References: &LT;3693fa55.17551686@news.intergate.bc.ca>
Newsgroups: sci.crypt
Lines: 36

In article &LT;3693fa55.17551686@news.intergate.bc.ca>,
  fred_vanandel@bigfoot.com wrote:
> I would like to thank those who spent time in answering my question.
> It has proven to be very helpful.
>
>  Terry Ritter wrote:
> >It might be interesting to know of an application where this sort of
> >precision is useful.
>
> The reason for the needed precision is that I am designing a  variable
> length hash function and I want to test it for resistance to
> collision.  Since I don't have the resources to do a full test  for a
> 256 bit hash I am going to test 8, 16, 24, 32 and maybe 40 bit hashes
> and search for trends. If the algorithm is resistant to collision in
> the smaller sizes and there is no trend away from the "proper" value
> then due to the nature of the algorithm I can be quite confidant that
> the larger hashes are also resistant to collisions.
>
> Fred Van Andel
>

  This is a very good idea. Since only if hash is any good will it
have the distribution predicticed by the bithday collision method.
However funny things do happen and though it is a very good idea
to check at the small lengths where more statistics can be made.
You should always run tests on the final lenght your going to use
or you might get surprised.

david scott

--
http://cryptography.org/cgi-bin/crypto.cgi/Misc/scott19u.zip
http://members.xoom.com/ecil/index.htm

-----------== Posted via Deja News, The Discussion Network ==----------
http://www.dejanews.com/       Search, Read, Discuss, or Start Your Own
```

```
Subject: Re: Birthday Attack calculations.
Date: Thu, 07 Jan 1999 03:53:13 GMT
From: fred_vanandel@bigfoot.com (Fred Van Andel)
Message-ID: &LT;36942dfe.1363360@news.intergate.bc.ca>
References: &LT;76vme0$avi$1@nnrp1.dejanews.com>
Newsgroups: sci.crypt
Lines: 32

>  fred_vanandel@bigfoot.com wrote:

> The reason for the needed precision is that I am designing a  variable
> length hash function and I want to test it for resistance to
> collision.  Since I don't have the resources to do a full test  for a
> 256 bit hash I am going to test 8, 16, 24, 32 and maybe 40 bit hashes
> and search for trends. If the algorithm is resistant to collision in
> the smaller sizes and there is no trend away from the "proper" value
> then due to the nature of the algorithm I can be quite confidant that
> the larger hashes are also resistant to collisions.

dscott@networkusa.net replied:

> This is a very good idea. Since only if hash is any good will it
>have the distribution predicticed by the bithday collision method.
>However funny things do happen and though it is a very good idea
>to check at the small lengths where more statistics can be made.
>You should always run tests on the final lenght your going to use
>or you might get surprised.

The whole point of testing on small hashes and extrapolating is that
is computationally impossible to do a birthday attack on a large hash.
For a 256 bit hash you will need to create more than  2^128 hashes
before the odds of a collision reach 50%.

Do you know how long that will take on my 486-66. Or even a planet
full of computers for that matter. The indirect evidence is the ONLY
indication of collision resistance.

Fred Van Andel
```

```
Subject: Re: Birthday Attack calculations.
Date: Thu, 07 Jan 1999 12:28:17 GMT
From: dscott@networkusa.net
Message-ID: &LT;7701g9$k15$1@nnrp1.dejanews.com>
References: &LT;36942dfe.1363360@news.intergate.bc.ca>
Newsgroups: sci.crypt
Lines: 59

In article &LT;36942dfe.1363360@news.intergate.bc.ca>,
  fred_vanandel@bigfoot.com wrote:
> >  fred_vanandel@bigfoot.com wrote:
>
> > The reason for the needed precision is that I am designing a  variable
> > length hash function and I want to test it for resistance to
> > collision.  Since I don't have the resources to do a full test  for a
> > 256 bit hash I am going to test 8, 16, 24, 32 and maybe 40 bit hashes
> > and search for trends. If the algorithm is resistant to collision in
> > the smaller sizes and there is no trend away from the "proper" value
> > then due to the nature of the algorithm I can be quite confidant that
> > the larger hashes are also resistant to collisions.
>
> dscott@networkusa.net replied:
>
> > This is a very good idea. Since only if hash is any good will it
> >have the distribution predicticed by the bithday collision method.
> >However funny things do happen and though it is a very good idea
> >to check at the small lengths where more statistics can be made.
> >You should always run tests on the final lenght your going to use
> >or you might get surprised.
>
> The whole point of testing on small hashes and extrapolating is that
> is computationally impossible to do a birthday attack on a large hash.
> For a 256 bit hash you will need to create more than  2^128 hashes
> before the odds of a collision reach 50%.
>
> Do you know how long that will take on my 486-66. Or even a planet
> full of computers for that matter. The indirect evidence is the ONLY
> indication of collision resistance.
>
> Fred Van Andel
>
>

 Again you should run tests on the final length. I did not say
to run the number necessary to get a 50% collision. It is more
than obvious you can't run that number of tests at your full
length. However it would be stupid not to run some tests in hopes
that no collision occur at long length. Though it some time appear at
all then something is wrong. So if it passes the short lengths
test still are needed for final length.
 The indirect evidence is NOT the ONLY indication you should
unless to urgent try the long case in hopes of no errors.

David Scott

P.S. Example I test scott16u very much. But I still had to
do a few tests on scott19u. And the upscaling produced a few
bugs that I would not have found and fixed if I smuggly
ignored testing all together on the longer version.

--
http://cryptography.org/cgi-bin/crypto.cgi/Misc/scott19u.zip
http://members.xoom.com/ecil/index.htm

-----------== Posted via Deja News, The Discussion Network ==----------
http://www.dejanews.com/       Search, Read, Discuss, or Start Your Own
```

```
Subject: Re: Birthday Attack calculations.
Date: Sat, 09 Jan 1999 00:14:20 GMT
From: dscott@networkusa.net
Message-ID: &LT;77671c$3d2$1@nnrp1.dejanews.com>
References: &LT;775qop$cbb@qualcomm.com>
   &LT;36942dfe.1363360@news.intergate.bc.ca>
Newsgroups: sci.crypt
Lines: 60

In article &LT;775qop$cbb@qualcomm.com>,
  ggr@qualcomm.com (Gregory G Rose) wrote:
> In article &LT;36942dfe.1363360@news.intergate.bc.ca>,
> Fred Van Andel &LT;fred_vanandel@bigfoot.com> wrote:
> &LT;dscott@networkusa.net replied:
> &LT;> This is a very good idea. Since only if hash is any good will it
> &LT;>have the distribution predicticed by the bithday collision method.
> &LT;>However funny things do happen and though it is a very good idea
> &LT;>to check at the small lengths where more statistics can be made.
> &LT;>You should always run tests on the final lenght your going to use
> &LT;>or you might get surprised.
> &LT;
> &LT;The whole point of testing on small hashes and extrapolating is that
> &LT;is computationally impossible to do a birthday attack on a large hash.
> &LT;For a 256 bit hash you will need to create more than  2^128 hashes
> &LT;before the odds of a collision reach 50%.
> &LT;
> &LT;Do you know how long that will take on my 486-66. Or even a planet
> &LT;full of computers for that matter. The indirect evidence is the ONLY
> &LT;indication of collision resistance.
>
> Now, let's not get testy. When David has a good
> idea, we should encourage it. The way I
> interpreted his comment was that, instead of
> testing the cut-down algorithms, cut down the
> output of the real algorithm. Generate your
> 256-bit hashes, split them into 32- or 64- bit
> chunks, and test *those* as if they had come from
> a smaller generator. Any collision problems in the
> larger hash, or any implementation glitch in
> scaling it up, should show up.
>
> regards,
> Greg.
> --

NO it is not even close to my thoughts

  I thought it was a very good reply on my part sorry
you where not capable of understanding it. So here
it is in different words. It was previously stated
that a cut down version of method tested. Apparently
short enough to be able get meaning full data based
on probability of collisions. But the thing was
worded that he should never test the FULL LENGTH
case since the collision probabilty was virtually
ZERO for any real number of tries one could make
with the FULL LENGTH case. But if the FULL LENGTH
case is what one is programming the stupid hash
function for. It would be FUCKING stupid not to run
a few cases just to be DAM SURE non occurred. People
can fuck up code you know.

David Scott

--
http://cryptography.org/cgi-bin/crypto.cgi/Misc/scott19u.zip
http://members.xoom.com/ecil/index.htm

-----------== Posted via Deja News, The Discussion Network ==----------
http://www.dejanews.com/       Search, Read, Discuss, or Start Your Own
```

```
Subject: Re: Birthday Attack calculations.
Date: Sat, 09 Jan 1999 05:44:15 GMT
From: fred_vanandel@bigfoot.com (Fred Van Andel)
Message-ID: &LT;3698df07.97942674@news.intergate.bc.ca>
References: &LT;77671c$3d2$1@nnrp1.dejanews.com>
Newsgroups: sci.crypt
Lines: 61

dscott@networkusa.net wrote:
>  ggr@qualcomm.com (Gregory G Rose) wrote:

>> Now, let's not get testy. When David has a good
>> idea, we should encourage it. The way I
>> interpreted his comment was that, instead of
>> testing the cut-down algorithms, cut down the
>> output of the real algorithm. Generate your
>> 256-bit hashes, split them into 32- or 64- bit
>> chunks, and test *those* as if they had come from
>> a smaller generator. Any collision problems in the
>> larger hash, or any implementation glitch in
>> scaling it up, should show up.
>>
>> regards,
>> Greg.
>> --
>
>NO it is not even close to my thoughts
>
>  I thought it was a very good reply on my part sorry
>you where not capable of understanding it. So here
>it is in different words. It was previously stated
>that a cut down version of method tested. Apparently
>short enough to be able get meaning full data based
>on probability of collisions. But the thing was
>worded that he should never test the FULL LENGTH
>case since the collision probabilty was virtually
>ZERO for any real number of tries one could make
>with the FULL LENGTH case. But if the FULL LENGTH
>case is what one is programming the stupid hash
>function for. It would be FUCKING stupid not to run
>a few cases just to be DAM SURE non occurred. People
>can fuck up code you know.
>
>David Scott
>
Unfortunately running one test of a 256 bit hash for even a tiny part
of its range will require far more disk space than I have access to
and will have virtually zero chance of finding a collision.  Even
testing a mere 2^32 hashes compared with the > 2^64 required for a 50%
collision of a 256 bit hash would require 128 GigaBytes of disk space
to store all of the calculated hashes. I only wish that I had that
much.

I will carry the testing into as far as I can go and still generate
statistically significant numbers. There is no point expending all my
computing power trying to force a collision of one relatively large
hash because the result would be a matter of dumb luck and completely
meaningless.  But by testing millions of tiny hashes and tens of
thousands of 40 or 48 bit hashes I can generate some meaningful
numbers.

If the hash is so fatally flawed that I could routinely find a
collision at much lower than the normal values then I would have found
this out at the smaller hash sizes.  The algorithm is scalable, there
is no real difference between a small hash and a large hash except for
the obvious.  Therefore I expect that the properties of the smaller
sizes to carry into the larger sizes.

Fred van Andel
```

```
Subject: Re: Birthday Attack calculations.
Date: Sat, 09 Jan 1999 13:40:58 GMT
From: dscott@networkusa.net
Message-ID: &LT;777m9a$985$1@nnrp1.dejanews.com>
References: &LT;3698df07.97942674@news.intergate.bc.ca>
Newsgroups: sci.crypt
Lines: 81

In article &LT;3698df07.97942674@news.intergate.bc.ca>,
  fred_vanandel@bigfoot.com wrote:
> dscott@networkusa.net wrote:

 snip...

>
> >NO it is not even close to my thoughts
> >
> >  I thought it was a very good reply on my part sorry
> >you where not capable of understanding it. So here
> >it is in different words. It was previously stated
> >that a cut down version of method tested. Apparently
> >short enough to be able get meaning full data based
> >on probability of collisions. But the thing was
> >worded that he should never test the FULL LENGTH
> >case since the collision probabilty was virtually
> >ZERO for any real number of tries one could make
> >with the FULL LENGTH case. But if the FULL LENGTH
> >case is what one is programming the stupid hash
> >function for. It would be FUCKING stupid not to run
> >a few cases just to be DAM SURE non occurred. People
> >can fuck up code you know.
> >
> >David Scott
> >
> Unfortunately running one test of a 256 bit hash for even a tiny part
> of its range will require far more disk space than I have access to
> and will have virtually zero chance of finding a collision.  Even
> testing a mere 2^32 hashes compared with the > 2^64 required for a 50%
> collision of a 256 bit hash would require 128 GigaBytes of disk space
> to store all of the calculated hashes. I only wish that I had that
> much.

   Then run 100 cases after your done with your low level
testing. I you can't run 100 cases your method sucks.
If you get any collisions at all in the 100 cases then you
FUCKED up the CODE. But you MUST check the CODE for some
cases that you actually suspect the code to run on.
  It reminds me of work when we bought code that was to run
on a DEC machine using VT100 terminals. The CRAP  didn't work
we talked to the company on phone several times. They claimed
it was developed on DEC machines using VT100 terminals after
months when one of there representatives came out to our site
after several visits they brought one of there VT100 terminals
the CRAP ran. But they had used a VT100 clone there code
would not work on a real VT100 DEC terminal. I think the
company died I sure hope so. But you have to run REAL TESTS
on the real one you expect the code to be used. Even if your
a pompous asshole and irrigantly expect no problem.

> I will carry the testing into as far as I can go and still generate
> statistically significant numbers. There is no point expending all my
> computing power trying to force a collision of one relatively large
> hash because the result would be a matter of dumb luck and completely
> meaningless.  But by testing millions of tiny hashes and tens of
> thousands of 40 or 48 bit hashes I can generate some meaningful
> numbers.
>
> If the hash is so fatally flawed that I could routinely find a
> collision at much lower than the normal values then I would have found
> this out at the smaller hash sizes.  The algorithm is scalable, there
> is no real difference between a small hash and a large hash except for
> the obvious.  Therefore I expect that the properties of the smaller
> sizes to carry into the larger sizes.

  You can expect all you want. Yor should more like a manager than
a real programmer with much common sense.

> Fred van Andel
>

David Scott

http://cryptography.org/cgi-bin/crypto.cgi/Misc/scott19u.zip
http://members.xoom.com/ecil/index.htm

-----------== Posted via Deja News, The Discussion Network ==----------
http://www.dejanews.com/       Search, Read, Discuss, or Start Your Own
```

```
Subject: Re: Birthday Attack calculations.
Date: Sun, 10 Jan 1999 11:49:11 GMT
From: dscott@networkusa.net
Message-ID: &LT;77a43m$7vu$1@nnrp1.dejanews.com>
References: &LT;369913dc.21716751@news.intergate.bc.ca>
   &LT;777m9a$985$1@nnrp1.dejanews.com>
Newsgroups: sci.crypt
Lines: 59

In article &LT;369913dc.21716751@news.intergate.bc.ca>,
  fred_vanandel@bigfoot.com wrote:
> dscott@networkusa.net wrote:
>
> >In article &LT;3698df07.97942674@news.intergate.bc.ca>,
> >  fred_vanandel@bigfoot.com wrote:
> >> dscott@networkusa.net wrote:
> >
> > snip...
> >
> >
> >    Then run 100 cases after your done with your low level
> >testing. I you can't run 100 cases your method sucks.
> >If you get any collisions at all in the 100 cases then you
> >FUCKED up the CODE. But you MUST check the CODE for some
> >cases that you actually suspect the code to run on.
> >  It reminds me of work when we bought code that was to run
> >on a DEC machine using VT100 terminals. The CRAP  didn't work
> >we talked to the company on phone several times. They claimed
> >it was developed on DEC machines using VT100 terminals after
> >months when one of there representatives came out to our site
> >after several visits they brought one of there VT100 terminals
> >the CRAP ran. But they had used a VT100 clone there code
> >would not work on a real VT100 DEC terminal. I think the
> >company died I sure hope so. But you have to run REAL TESTS
> >on the real one you expect the code to be used. Even if your
> >a pompous asshole and irrigantly expect no problem.
> >
>
> There is not enough computing power on the planet  to do a full
> collision test on a 256 bit hash.  Do the math yourself.
>
> >  You can expect all you want. Yor should more like a manager than
> >a real programmer with much common sense.
>
> I think there is an insult in there somewhere, I guess that means I
> have arrived.
>

  Actual I am disapointed that some one else caught your mathematical
erros while I was just wasting my time telling you over and over
that a FULL collision test on your 256 bit hash was not needed.
But you do need to try a few cases just to verify that ZERO occur
but your obviously to pigheaded to understand this point. If
this makes you think you have arrived then go ahead your arrived
just like mr H.

> Fred Van Andel
>

David Scott
P.S. I never did trust the stability of a Van compared to
a Car.

http://cryptography.org/cgi-bin/crypto.cgi/Misc/scott19u.zip
http://members.xoom.com/ecil/index.htm

-----------== Posted via Deja News, The Discussion Network ==----------
http://www.dejanews.com/       Search, Read, Discuss, or Start Your Own
```

```
Subject: Re: Birthday Attack calculations.
Date: Sat, 09 Jan 1999 04:46:16 GMT
From: fred_vanandel@bigfoot.com (Fred Van Andel)
Message-ID: &LT;3696dc0f.97390149@news.intergate.bc.ca>
References: &LT;775qop$cbb@qualcomm.com>
   &LT;36942dfe.1363360@news.intergate.bc.ca>
Newsgroups: sci.crypt
Lines: 27

ggr@qualcomm.com (Gregory G Rose) wrote:

>Now, let's not get testy. When David has a good
>idea, we should encourage it. The way I
>interpreted his comment was that, instead of
>testing the cut-down algorithms, cut down the
>output of the real algorithm. Generate your
>256-bit hashes, split them into 32- or 64- bit
>chunks, and test *those* as if they had come from
>a smaller generator. Any collision problems in the
>larger hash, or any implementation glitch in
>scaling it up, should show up.

The smaller hash sizes are not from a 'cut-down' algorithm. The nature
of this algorithm is that is can create hashes of any desired size
(actually in 8 bit multiples). Any given size is no more 'natural'
than any other size.  If a smaller hash is collision resistant then I
would expect that a larger hash is also collision resistant. The only
real difference is that the small hashes are very easy to test, the
larger one are more difficult.

I hope to have access to about 12 Pentium machines during evenings and
weekends for testing purposes,  this will allow me to crunch quite a
few samples in search of collisions.

Fred Van Andel
```

```
Subject: Re: Birthday Attack calculations.
Date: Sat, 09 Jan 1999 15:20:47 -0500
From: "Trevor Jackson, III" &LT;fullmoon@aspi.net>
Message-ID: &LT;3697BA1F.9FE69816@aspi.net>
References: &LT;36942dfe.1363360@news.intergate.bc.ca>
Newsgroups: sci.crypt
Lines: 31

> The whole point of testing on small hashes and extrapolating is that
> is computationally impossible to do a birthday attack on a large hash.
> For a 256 bit hash you will need to create more than  2^128 hashes
> before the odds of a collision reach 50%.
>
> Fred Van Andel

Something is wrong with this paragraph.  Either the attack is not a birthday
attack, or the odds of a collision are much higher.  An exhaustive search of
a 2^256 element state space has a 50% chance of a match against the target of
the search.  This is not a birthday attack.

A birthday attack searches for any matching pair rather than a match against
a target.  Thus a birthday attack searching N states implies generating O(N)
states, but comparisons among N states means O(N^2) comparisons (assuming
the states are not generated in an order that allows comparison against
neighbors only).  Thus for N small the generation cost may dominate the
overall cost of the search but for even a 40 bit key the comparison cost
will dominate due to the 2^79 comparisons required.

The odds of finding a matching pair among N of M states is the product over
0 to N-1 of the expression (M-i)/M.

    odds = 1;
    for i=0 to N-1
        odds = odds * (M-i)/M

Thus a birthday attack can find a matching pair of elements much faster than
an exhaustive search can find a match for a chosen target.
```

```
Subject: Re: Birthday Attack calculations.
Date: Sun, 10 Jan 1999 08:31:51 -0500
From: "Trevor Jackson, III" &LT;fullmoon@aspi.net>
Message-ID: &LT;3698A0C7.B01FDDD9@aspi.net>
References: &LT;3697c1c9.7030828@news.intergate.bc.ca>
   &LT;3697BA1F.9FE69816@aspi.net>
Newsgroups: sci.crypt
Lines: 80

Fred Van Andel wrote:

> "Trevor Jackson, III" &LT;fullmoon@aspi.net> wrote:
>
> >> The whole point of testing on small hashes and extrapolating is that
> >> is computationally impossible to do a birthday attack on a large hash.
> >> For a 256 bit hash you will need to create more than  2^128 hashes
> >> before the odds of a collision reach 50%.
> >>
> >> Fred Van Andel
> >
> >Something is wrong with this paragraph.  Either the attack is not a birthday
> >attack, or the odds of a collision are much higher.  An exhaustive search of
> >a 2^256 element state space has a 50% chance of a match against the target of
> >the search.  This is not a birthday attack.
> >
> >A birthday attack searches for any matching pair rather than a match against
> >a target.  Thus a birthday attack searching N states implies generating O(N)
> >states, but comparisons among N states means O(N^2) comparisons (assuming
> >the states are not generated in an order that allows comparison against
> >neighbors only).  Thus for N small the generation cost may dominate the
> >overall cost of the search but for even a 40 bit key the comparison cost
> >will dominate due to the 2^79 comparisons required.
> >
> >The odds of finding a matching pair among N of M states is the product over
> >0 to N-1 of the expression (M-i)/M.
```

```
> >
> >    odds = 1;
> >    for i=0 to N-1
> >        odds  = odds * (M-i)/M
> >
> >Thus a birthday attack can find a matching pair of elements much faster than
> >an exhaustive attack can find a match for a chosen target.
>
> Lets see if I understand you correctly. First you generate a pair of
> hashes and compare them to each other.  If they don't match then you
> throw them away and generate a new pair. Repeat until you find a
> match.
>

Hardly.  Throwing away samples wastes the effort invested in creating them.  For
the purpose of the reply I assumed that each sample would be stored after
comparison with the previously stored samples.  Thus we need to store N samples,
and compare (N^2)/2 times.

> But by this method each pair only has one chance in N of being a
> collision and therefore would require N/2 total comparisons to reach
> the 50% level.  When we are talking large hashes this becomes a hugh
> number.
>
> Another way is to generate all the hashes up front, sort them and then
> compare.  For example with a 64 bit hash you could generate your 1.2 *
> 2^32 hashes and save them to a file.  Sorting the file will take
> roughly 32 * 2^32 compares for a total of approx 2^37 operations.

> This is much more feasible that the 2^63 operations required when

> testing pairs of data.  You could run approximately 2^26 of the
> sorting tests in the same time as it takes to run one of the pairs
> test.
>
> Bear in mind that this method does not give you the average but rather
> how often you are successful at the 50% level.  This is a slightly
> different number.  If you want the true average you must generate more
> numbers and then determine afterwards where the match occurred.
>
> You refer to 2^79 compares required for a 40 bit hash.  This the
> absolute worst case in which no collisions are generated even after
> all possible hashes have been created.  The odds of reaching the end
> of the hash without a collision are extremely remote. Even for a hash
> as small as 8 bits, the odds of generating all possible hashes without
> a collision is about 1 in 10^110.  For our 40 bit example most of the
> cases will cluster around 20 bits that corresponds to about 2^39
> compares. That is much more manageable.

You are still using the numbers for exhaustive search.  The birthday attack
numbers are much smaller.  As I don't have an arbitrary precision calculator at
hand I'll have to do some work to generate them accurately.  I'll try to generate
them and post here
```

```
Subject: Re: Birthday Attack calculations.
Date: Mon, 11 Jan 1999 03:43:01 GMT
From: fred_vanandel@bigfoot.com (Fred Van Andel)
Message-ID: &LT;36996c54.7217058@news.intergate.bc.ca>
References: &LT;3698ABC7.B01FDDD9@aspi.net>
Newsgroups: sci.crypt
Lines: 26

"Trevor Jackson, III" &LT;fullmoon@aspi.net> wrote:

>Hardly.  Throwing away samples wastes the effort invested in creating them.  For
>the purpose of the reply I assumed that each sample would be stored after
>comparison with the previously stored samples.

I obviously misunderstood what you were trying to say and I tailored
my comments around that misunderstanding.   My apologies for
attempting to put words in your mouth.

However I still stand by my original statement.  A birthday attack on
a 256 bit hash would require in excess of 2^128 hashes to be
calculated and stored before the odds of a collision reach 50%.

>You are still using the numbers for exhaustive search.  The birthday attack
>numbers are much smaller.  As I don't have an arbitrary precision calculator at
>hand I'll have to do some work to generate them accurately.  I'll try to generate
>them and post here.

A birthday attack would require > 2^128 calculation while an
exhaustive search would need 2^255.   There is a big difference. When
you are dealing with large hashes even a birthday attack becomes
impossible to do.  Its the birthday attach that dictates the size of
hash required, not the exhaustive search.

Fred Van Andel
```

```
Subject: Re: Birthday Attack calculations.
Date: Mon, 11 Jan 1999 22:45:19 -0500
From: "Trevor Jackson, III" &LT;fullmoon@aspi.net>
Message-ID: &LT;369AC54F.CE51AD60@aspi.net>
References: &LT;36996c54.7217058@news.intergate.bc.ca>
Newsgroups: sci.crypt
Lines: 12

Fred Van Andel wrote:

> A birthday attack would require > 2^128 calculation while an
> exhaustive search would need 2^255.   There is a big difference. When
> you are dealing with large hashes even a birthday attack becomes
> impossible to do.  Its the birthday attach that dictates the size of
> hash required, not the exhaustive search.

I agree with your last statement above.  But I an confused by your first statement
above.  Is there a closed-form equation for the figure 2^128 you quoted?  I am only
familiar with the probability series summation.
```

```
Subject: Re: Birthday Attack calculations.
Date: Tue, 12 Jan 1999 06:34:34 GMT
From: fred_vanandel@bigfoot.com (Fred Van Andel)
Message-ID: &LT;369ae5c4.103855861@news.intergate.bc.ca>
References: &LT;369AC54F.CE51AD60@aspi.net>
Newsgroups: sci.crypt
Lines: 97

"Trevor Jackson, III" &LT;fullmoon@aspi.net> wrote:

>Fred Van Andel wrote:
>
>> A birthday attack would require > 2^128 calculation while an
>> exhaustive search would need 2^255.   There is a big difference. When
>> you are dealing with large hashes even a birthday attack becomes
>> impossible to do.  Its the birthday attach that dictates the size of
>> hash required, not the exhaustive search.
>
>I agree with your last statement above.  But I an confused by your first statement
>above.  Is there a closed-form equation for the figure 2^128 you quoted?  I am only
>familiar with the probability series summation.

You quoted the formula yourself in a earlier message

> odds = 1;
>    for i=0 to N-1
>        odds  = odds * (M-i)/M

For any gives value of M the location of the 50% mark will be roughly
the square root of M.  The square root of 2^256 is 2^128.

Unless I am misunderstanding you again.

Some closed forms of the equations were given in the earlier posts of
this thread.

This method was posted by David Broughton

>    w = n^g + 0.29 - e
>
>where:
>w = the number of documents to get 50% probability of a collision
>n = number of different hash codes, all equiprobable
>g = 0.5 + 1/(6.13 * ln(n))
>ln() is the natural logarithm function
>e is a small error that can be ignored in practice, usually &LT; +- 1.
>
>This is an empirical formula that I worked out many years ago and
>filed away in a notebook.
>
>The exact formula is the value of w in this equation:
>
>   ( product from k = 1 to k = w-1 of (n-k)/n ) = 0.5
>
>but this is not a practical calculation for large n.
>
>As you can see, w is a bit larger than the square root of n.  For
>n = 10^6, for example, w = 1177.68 (e = -0.197).
>
>If your formula is meant to be 1.2 * n^0.5, then w = 1200 for n =
>10^6 which is getting there.

Or this one by Peter Pearson

>To derive your own approximation formula from the above, exact
>formula, rewrite it as follows:
>
>           n (n-1) (n-2) ... (n+1-w)       n!
>product = ------------------------- = ----------
>                   n^w                (n-w)! n^w
>
>Then, use Stirling's approximation to make the factorials more
>manageable. Stirling's approximation (see, for example, Knuth,
>Art of Computer Programming, Volume 1) is:
>
> ln( n! ) = (n+1/2) ln(n) - n + ln( 2*pi )/2 + 1/(12*n) - ...
>
>You'll have to experiment with the number of terms required to
>get meaningful results. Overimplifying to n*ln(n)-n gives conspicuously
>nonsensical results. If memory serves, (n+1/2) ln(n) - n + ln( 2*pi )/2
>is a good level of approximation, and one needs the approximation
>ln(1+x) = x (for small x) as well.

Or this one by Terry Ritter

>   s(N,p) = (1 + SQRT(1 - 8N ln(p))) / 2
>
>where s is the expected number of samples needed, N the size of the
>population being sampled, and p the given probability.
>
>For N = 10**6 and p = 0.5 (so ln(p) = -0.693) we get 1177.9 instead of
>the usual handwave SQRT(N) = 1000, or the stated approximation, 1.2 *
>SQRT(N) = 1200.
>
>For N = 2**20 and p = 0.5, we get 1206.2 instead of 1.2 * 1024 =
>1228.8 for the stated approximation.
>
>The formula basically comes out of my article on population
>estimation:
>
>  http://www.io.com/~ritter/ARTS/BIRTHDAY.HTM
>

Fred Van Andel
```

```
Subject: Re: Birthday Attack calculations.
Date: Tue, 12 Jan 1999 08:34:48 -0500
From: "Trevor Jackson, III" &LT;fullmoon@aspi.net>
Message-ID: &LT;369B4F77.E076C774@aspi.net>
References: &LT;369ae5c4.103855861@news.intergate.bc.ca>
Newsgroups: sci.crypt
Lines: 112

Thanks for the background exposition.  I think I am slightly confused over the odds
of
finding the first collision and the odds of finding no or any  collisions.

For the purposes of the original issue we want the odds of finding the first
collision.
I.e., the expected wait.  The series formula I described indicates the probability of
finding no collisions, which also gives (by 1-p) the probability of finding any
number
of collisions.

Presumably even odds of finding a single collision should take less work than even
odds
of finding all collisions.  Is this correct?

Fred Van Andel wrote:

> "Trevor Jackson, III" &LT;fullmoon@aspi.net> wrote:
>
> >Fred Van Andel wrote:
> >
> >> A birthday attack would require > 2^128 calculation while an
> >> exhaustive search would need 2^255.   There is a big difference. When
> >> you are dealing with large hashes even a birthday attack becomes
> >> impossible to do.  Its the birthday attach that dictates the size of
> >> hash required, not the exhaustive search.
> >
> >I agree with your last statement above.  But I an confused by your first statement
> >above.  Is there a closed-form equation for the figure 2^128 you quoted?  I am
only
> >familiar with the probability series summation.
>
> You quoted the formula yourself in a earlier message
>
> > odds = 1;
> >    for i=0 to N-1
> >        odds  = odds * (M-i)/M
>
> For any gives value of M the location of the 50% mark will be roughly
> the square root of M.  The square root of 2^256 is 2^128.
>
> Unless I am misunderstanding you again.
>
> Some closed forms of the equations were given in the earlier posts of
> this thread.
>
> This method was posted by David Broughton
>
> >    w = n^g + 0.29 - e
> >
> >where:
> >w = the number of documents to get 50% probability of a collision
> >n = number of different hash codes, all equiprobable
> >g = 0.5 + 1/(6.13 * ln(n))
> >ln() is the natural logarithm function
> >e is a small error that can be ignored in practice, usually &LT; +- 1.
> >
> >This is an empirical formula that I worked out many years ago and
> >filed away in a notebook.
> >
> >The exact formula is the value of w in this equation:
> >
> >   ( product from k = 1 to k = w-1 of (n-k)/n ) = 0.5
> >
> >but this is not a practical calculation for large n.
> >
> >As you can see, w is a bit larger than the square root of n.  For
> >n = 10^6, for example, w = 1177.68 (e = -0.197).
> >
> >If your formula is meant to be 1.2 * n^0.5, then w = 1200 for n =
> >10^6 which is getting there.
>
> Or this one by Peter Pearson
>
> >To derive your own approximation formula from the above, exact
> >formula, rewrite it as follows:
> >
> >           n (n-1) (n-2) ... (n+1-w)       n!
> >product = ------------------------- = ----------
> >                   n^w                (n-w)! n^w
> >
> >Then, use Stirling's approximation to make the factorials more
> >manageable. Stirling's approximation (see, for example, Knuth,
> >Art of Computer Programming, Volume 1) is:
> >
> > ln( n! ) = (n+1/2) ln(n) - n + ln( 2*pi )/2 + 1/(12*n) - ...
> >
> >You'll have to experiment with the number of terms required to
> >get meaningful results. Overimplifying to n*ln(n)-n gives conspicuously
> >nonsensical results. If memory serves, (n+1/2) ln(n) - n + ln( 2*pi )/2
> >is a good level of approximation, and one needs the approximation
> >ln(1+x) = x (for small x) as well.
>
> Or this one by Terry Ritter
>
> >   s(N,p) = (1 + SQRT(1 - 8N ln(p))) / 2
> >
> >where s is the expected number of samples needed, N the size of the
> >population being sampled, and p the given probability.
> >
> >For N = 10**6 and p = 0.5 (so ln(p) = -0.693) we get 1177.9 instead of
> >the usual handwave SQRT(N) = 1000, or the stated approximation, 1.2 *
> >SQRT(N) = 1200.
> >
> >For N = 2**20 and p = 0.5, we get 1206.2 instead of 1.2 * 1024 =
> >1228.8 for the stated approximation.
> >
> >The formula basically comes out of my article on population
> >estimation:
> >
> >  http://www.io.com/~ritter/ARTS/BIRTHDAY.HTM
> >
>
> Fred Van Andel
```

```
Subject: Re: Birthday Attack calculations.
Date: Wed, 13 Jan 1999 06:50:51 GMT
From: fred_vanandel@bigfoot.com (Fred Van Andel)
Message-ID: &LT;369d3d9c.8943462@news.intergate.bc.ca>
References: &LT;369B4F77.E076C774@aspi.net>
Newsgroups: sci.crypt
Lines: 27

"Trevor Jackson, III" &LT;fullmoon@aspi.net> wrote:

>Thanks for the background exposition.  I think I am slightly confused over the odds
of
>finding the first collision and the odds of finding no or any  collisions.
>
>For the purposes of the original issue we want the odds of finding the first
collision.
>I.e., the expected wait.  The series formula I described indicates the probability
of
>finding no collisions, which also gives (by 1-p) the probability of finding any
number
>of collisions.
>
>Presumably even odds of finding a single collision should take less work than even
odds
>of finding all collisions.  Is this correct?

Using the equasion below the AVERAGE number of hashes that need to be
tested is calculated by:
   i = 1'
   odds = 1;
   M = Whatever;
   while( odds > 0.5)
      {
         odds  = odds * (M-i)/M;
         i++;
      }

For a value of 1,000,000, i is 1178.

Fred Van Andel
```

```
Subject: Re: Birthday Attack calculations.
Date: Wed, 13 Jan 1999 07:52:27 GMT
From: ritter@io.com (Terry Ritter)
Message-ID: &LT;369c50a1.3513345@news.io.com>
References: &LT;369d3d9c.8943462@news.intergate.bc.ca>
Newsgroups: sci.crypt
Lines: 35


On Wed, 13 Jan 1999 06:50:51 GMT, in
&LT;369d3d9c.8943462@news.intergate.bc.ca>, in sci.crypt
fred_vanandel@bigfoot.com (Fred Van Andel) wrote:

>[...]
>Using the equasion below the AVERAGE number of hashes that need to be
>tested is calculated by:
>   i = 1'
>   odds = 1;
>   M = Whatever;
>   while( odds > 0.5)
>      {
>         odds  = odds * (M-i)/M;
>         i++;
>      }
>
>For a value of 1,000,000, i is 1178.

Which seems to compare rather well to the value 1177.9 computed from
my formula in the earlier posting:

|   s(N,p) = (1 + SQRT(1 - 8N ln(p))) / 2
|
|where s is the expected number of samples needed, N the size of
|the population being sampled, and p the given probability.
|
|For N = 10**6 and p = 0.5 (so ln(p) = -0.693) we get 1177.9 instead
|of the usual handwave SQRT(N) = 1000 [...]

---
Terry Ritter   ritter@io.com   http://www.io.com/~ritter/
Crypto Glossary   http://www.io.com/~ritter/GLOSSARY.HTM
```

```
Subject: Re: Birthday Attack calculations.
Date: Thu, 14 Jan 1999 04:37:24 GMT
From: fred_vanandel@bigfoot.com (Fred Van Andel)
Message-ID: &LT;369d73bf.1141606@news.intergate.bc.ca>
References: &LT;369c50a1.3513345@news.io.com>
Newsgroups: sci.crypt
Lines: 9

ritter@io.com (Terry Ritter) wrote:
>>For a value of 1,000,000, i is 1178.
>
>Which seems to compare rather well to the value 1177.9 computed from
>my formula in the earlier posting:

Why do you think I chose the number 1,000,000.  Its a good idea to
check your answers before making a fool of yourself in front of the
multitude.
```

Terry Ritter, his current address, and his top page.

Last updated: 1999-02-20

Tests for Randomness

A *Ciphers By Ritter* Page

A discussion about distinguishing a random sequence from a non-random sequence.

Contents

- 1998-11-14 Scott Nelson: "There is no known way to prove randomness. There's a pretty strong argument that randomness can't even be tested properly."
- 1998-11-15 Douglas A. Gwyn: "What *can* be done (in many cases) is to measure the information in favor of some hypothesis (about the source) over some other hypothesis...."
- 1998-11-16 Mok-Kong Shen: "For random bit sequences I recommend Maurer's test...."
- 1998-11-16 Andrew: "The tests I was most interested in would also include an amplitude (distance) or a two dimensional space."
- 1998-11-16 Scott Nelson: "Some of the tests in the Diehard battery test in 2, 3, 4 and 5 dimensions."
- 1998-11-17 Douglas A. Gwyn: "If that pattern continues as the sequence is extended, then the sequence is certainly *not* random."
- 1998-12-10 bob_jenkins@my-dejanews.com: "The test for that is called the 'run test'."
- 1998-12-11 Terry Ritter: "In my experience, Runs-Up and Runs-Down are far more powerful than they may appear."
- 1998-12-11 Tony T. Warnock: "I've found the same thing."

```
Subject: Re: Tests for Randomness
Date: Sat, 14 Nov 1998 17:41:05 GMT
From: scott@helsbreth.org (Scott Nelson)
Message-ID: &LT;3652bce8.168156049@news.inreach.com>
References: &LT;364C7F53.9F43FCA3@aspco.com>
Newsgroups: sci.crypt
Lines: 30

On Fri, 13 Nov 1998 13:49:55 -0500, Andrew &LT;Andrew@aspco.com> wrote:

>Can anyone e-mail me some information on how one can best analyze a set
>of data to dtermine if it is a random sequence or non random one?  Any
>help would be greatly appreciated.
>
Short answer:
No.

Long answer:
There is no known way to prove randomness.
There's a pretty strong argument that randomness can't even
be tested properly.  And the (poor) tests we have are really
only useful when applied to very large sets.  As a rule of
thumb, you need about n squared data points to achieve
a confidence of 1/n in the results (which shouldn't be
confused with confidence in the randomness of the generator)
Some of the tests in Diehard use over 2 million 32 bit
samples.

Still, applying the tests we have, as bad as they are,
is better than nothing.  Take a look at
http://stat.fsu.edu/~geo/diehard.html
(If you're a C programmer, you might prefer DiehardC -
ftp://helsbreth.org/pub/helsbret/random )

-----------------------------------
Shameless plug for random web site:
http://www.helsbreth.org/random
Scott Nelson &LT;scott@helsbreth.org>
```

```
Subject: Re: Tests for Randomness
Date: Sun, 15 Nov 1998 07:26:57 GMT
From: "Douglas A. Gwyn" &LT;DAGwyn@null.net>
Message-ID: &LT;364E8229.5584A45@null.net>
References: &LT;3652bce8.168156049@news.inreach.com>
Newsgroups: sci.crypt
Lines: 13

Scott Nelson wrote:
> There's a pretty strong argument that randomness can't even
> be tested properly.

What *can* be done (in many cases) is to measure the information
in favor of some hypothesis (about the source) over some other
hypothesis; for example, that the observed sequence was generated
by a uniform-random source rather than a source with certain
other specified characteristics.

That changes an ill-posed question "Is it random?" into an
answerable question where the answer has useful mathematical
properties.
```

```
Subject: Re: Tests for Randomness
Date: Mon, 16 Nov 1998 19:33:47 +0100
From: Mok-Kong Shen &LT;mok-kong.shen@stud.uni-muenchen.de>
Message-ID: &LT;3650700B.A1122213@stud.uni-muenchen.de>
References: &LT;364C7F53.9F43FCA3@aspco.com>
Newsgroups: sci.crypt
Lines: 11

Andrew wrote:
>
> Can anyone e-mail me some information on how one can best analyze a set
> of data to dtermine if it is a random sequence or non random one?  Any
> help would be greatly appreciated.

For random bit sequences I recommend Maurer's test which is described
in Menezes et al., Handbook of Applied Cryptography. It is very simple
to implement. I have a Fortran code for that.

M. K. Shen
```

```
Subject: Re: Tests for Randomness
Date: Mon, 16 Nov 1998 14:17:33 -0500
From: Andrew &LT;Andrew@aspco.com>
Message-ID: &LT;36507A4D.503FCABC@aspco.com>
References: &LT;364C7F53.9F43FCA3@aspco.com>
Newsgroups: sci.crypt
Lines: 54

I appreciate all those who have responded to my original post.  While I
have not had the opportunity to examine in detail all of the information
and related web sites, I have noticed one thing.  I will attempt to
explain, to the best of my abilities, a question.  Please forgive me for
any incorrect use of terminology.

A majority of the tests for randomness which were recommended to me,
dealt primarily with sequencing or one dimensional space.  The tests I
was most interested in would also include an amplitude (distance) or a
two dimensional space.  For example:

Suppose you have the following sequence:    5 -1 4 -2 6 -1 5 -2 4 0 5 -1

because it generally follows: positive, negative, positive, negative
.... it might appear that this is a randomly distributed set of values.
The graph would look like this:



Looking more closely, if you looked at the graph of the result of the
data it would look like this:



With this graph you can clearly see a "trend" in the data that is
present.  Using the average "step size" and number of iterations one
could calculate the randomness of where you ended up from where you
started.

I was hoping that people on this list might know of additional tests
which would take that set of data and indicate if it was random or not.

Again, Any advice or assistance is greatly appreciated.  I look forward
to hearing from you all.  Thanks for your time.

Andrew


Andrew wrote:

> Can anyone e-mail me some information on how one can best analyze a
> set
> of data to dtermine if it is a random sequence or non random one?  Any
>
> help would be greatly appreciated.
>
> Thanks for your time.
>
> Regards,
>
> Andrew
> andrew@aspco.com
```

```
Subject: Re: Tests for Randomness
Date: Mon, 16 Nov 1998 21:45:28 GMT
From: scott@helsbreth.org (Scott Nelson)
Message-ID: &LT;36508f0f.11893416@news.inreach.com>
References: &LT;36507A4D.503FCABC@aspco.com>
Newsgroups: sci.crypt
Lines: 39

On Mon, 16 Nov 1998 14:17:33 -0500, Andrew &LT;Andrew@aspco.com> wrote:

>I appreciate all those who have responded to my original post.  While I
>have not had the opportunity to examine in detail all of the information
>and related web sites, I have noticed one thing.  I will attempt to
>explain, to the best of my abilities, a question.  Please forgive me for
>any incorrect use of terminology.
>
>A majority of the tests for randomness which were recommended to me,
>dealt primarily with sequencing or one dimensional space.
>
[example snipped]

>I was hoping that people on this list might know of additional tests
>which would take that set of data and indicate if it was random or not.
>
>Again, Any advice or assistance is greatly appreciated.  I look forward
>to hearing from you all.  Thanks for your time.
>

I think you need to learn a bit more of the nomenclature before
you ask again.  If I roll a die 10 times and get the sequence
1,2,3,4,5,6,6,6,6,6 that sequence is just as random as the
sequence 5,2,3,5,1,2,4,6,2,2 or any other, but I suspect
you would reject the first as "not random enough"

Some of the tests in the Diehard battery test in 2, 3, 4 and 5
dimensions.  _Any_ sequence generated by a pseudo random
generator will fail to be random when viewed in enough dimensions,
though "enough" might be larger than is practical.  For instance,
a cryptographically secure pseudo random number generator shouldn't
fail until the number of dimensions is greater than the key space.

Not sure if that answers your question or not, but I hope it helps.

---------------------------------------
DiehardC 1.03 now available via ftp from
ftp://helsbreth.org/pub/helsbret/random
Scott Nelson &LT;scott@helsbreth.org>
```

```
Subject: Re: Tests for Randomness
Date: Tue, 17 Nov 1998 10:02:50 GMT
From: "Douglas A. Gwyn" &LT;DAGwyn@null.net>
Message-ID: &LT;365149A3.E3F4031F@null.net>
References: &LT;36507A4D.503FCABC@aspco.com>
Newsgroups: sci.crypt
Lines: 8

Andrew wrote:
> Suppose you have the following sequence:    5 -1 4 -2 6 -1 5 -2 4 0 5 -1
> because it generally follows: positive, negative, positive, negative
> .... it might appear that this is a randomly distributed set of values.

You need to distinguish between the set and the sequence.
If that pattern continues as the sequence is extended,
then the sequence is certainly *not* random.
```

```
Subject: Re: Tests for Randomness
Date: Thu, 10 Dec 1998 23:39:02 GMT
From: bob_jenkins@my-dejanews.com
Message-ID: &LT;74pm2m$9nn$1@nnrp1.dejanews.com>
References: &LT;36507A4D.503FCABC@aspco.com>
Newsgroups: sci.crypt
Lines: 17

In article &LT;36507A4D.503FCABC@aspco.com>,
  Andrew &LT;Andrew@aspco.com> wrote:

> Suppose you have the following sequence:    5 -1 4 -2 6 -1 5 -2 4 0 5 -1
>
> because it generally follows: positive, negative, positive, negative
> .... it might appear that this is a randomly distributed set of values.

The test for that is called the "run test".  It measures how long a run of
strictly increasing values you have.  It's one of the standard tests, not a
particularly powerful one though.

- Bob Jenkins
http://ourworld.compuserve.com/homepages/bob_jenkins

-----------== Posted via Deja News, The Discussion Network ==----------
http://www.dejanews.com/       Search, Read, Discuss, or Start Your Own
```

```
Subject: Re: Tests for Randomness
Date: Fri, 11 Dec 1998 05:15:11 GMT
From: ritter@io.com (Terry Ritter)
Message-ID: &LT;3670a993.3798156@news.io.com>
References: &LT;74pm2m$9nn$1@nnrp1.dejanews.com>
Newsgroups: sci.crypt
Lines: 37


On Thu, 10 Dec 1998 23:39:02 GMT, in
&LT;74pm2m$9nn$1@nnrp1.dejanews.com>, in sci.crypt
bob_jenkins@my-dejanews.com wrote:

>In article &LT;36507A4D.503FCABC@aspco.com>,
>  Andrew &LT;Andrew@aspco.com> wrote:
>
>> Suppose you have the following sequence:    5 -1 4 -2 6 -1 5 -2 4 0 5 -1
>>
>> because it generally follows: positive, negative, positive, negative
>> .... it might appear that this is a randomly distributed set of values.
>
>The test for that is called the "run test".  It measures how long a run of
>strictly increasing values you have.

That sounds like "Runs-Up."


>It's one of the standard tests, not a
>particularly powerful one though.

In my experience, Runs-Up and Runs-Down are far more powerful than
they may appear.  In contrast to many other randomness tests, Runs-Up
and Runs-Down can expose correlations (errors of conditional
probability), to some extent.

For small value ranges it is necessary to use the exact formulation
for the expectations, which is not in Knuth II.  See:

   http://www.io.com/~ritter/ARTS/RUNSUP.HTM

---
Terry Ritter   ritter@io.com   http://www.io.com/~ritter/
Crypto Glossary   http://www.io.com/~ritter/GLOSSARY.HTM
```

```
Subject: Re: Tests for Randomness
Date: Fri, 11 Dec 1998 08:12:06 -0700
From: "Tony T. Warnock" &LT;u091889@cic-mail.lanl.gov>
Message-ID: &LT;36713646.6494E470@cic-mail.lanl.gov>
References: &LT;3670a993.3798156@news.io.com>
Newsgroups: sci.crypt
Lines: 48


Terry Ritter wrote:.

> In my experience, Runs-Up and Runs-Down are far more powerful than
> they may appear.  In contrast to many other randomness tests, Runs-Up
> and Runs-Down can expose correlations (errors of conditional
> probability), to some extent.
>
> For small value ranges it is necessary to use the exact formulation
> for the expectations, which is not in Knuth II.  See:
>
>    http://www.io.com/~ritter/ARTS/RUNSUP.HTM
>

I've found the same thing. These are also fairly easy to implement.


Tony
```

Terry Ritter, his current address, and his top page.

Last updated: 1999-01-19

Bit Counting and Similar Instructions

A Cipher by Ritter Page

Contents

- 1000-CE/Cray/Ibm: "Ones in bytes with double-addition-tagged instructions for ala: the following (example-related) functions:
  - Counts the number of ones in a word
  - Counts the number of ones in a word, and also tells if it is a power of two
  - Reverse the positions of the bits significant 1 in a word
  - Reverse the positions of the bits in a word
- 1000-CE/Cray/Ibm: "Ones in bytes with double-addition-tagged instructions for ala: the following (example-related) functions:
- 1000-CE/Ibm/Mips

Terje
--
- <Terje.Mathisen@hda.hydro.com>
Using self-discipline, see http://www.eiffel.com/discipline
"almost all programming can be viewed as an exercise in caching"

Originator: nmm1@taurus.cus.cam.ac.uk

Subject: Re: Bit counting and similar instructions
Date: 19 Jan 1999 12:55:43 GMT
From: nmm1@cus.cam.ac.uk (Nick Maclaren)
Message-ID: <781vcfsiu1@pegasus.csx.cam.ac.uk>
References: <36A47BD3.804DF5B0@hda.hydro.com>
Newsgroups: comp.arch.arithmetic
Lines: 30

In article <36A47BD3.804DF5B0@hda.hydro.com>, Terje Mathisen
<Terje.Mathisen@hda.hydro.com> writes:

```
|> Christian Bau wrote:
|> > There is no single instruction to count bits in a word on the PowerPC,
|> > only an instruction to count the number of leading zero bits. Written in
|> > C, the following code would be quite efficient on the PowerPC if you do
|> > the operation a lot:  ...
|> >
|> > Total latency six cycles if a pointer to the lookup table is in a
|>
|> A Pentium would be similar:  ...
|>
|> > .... I think the Altivec processors (PowerPC 750 + Vector
|> > instructions) could be quite fast at it as well. My guess is about 5
|> > cycles for counting 128 bits.
```

Most machines are similar to the above. The benefits of such an instruction would be marginal, at best, given the small number of programs that would benefit.

On the other hand, the cost of implementing such an instruction is usually also marginal ....

Regards,
Nick Maclaren,
University of Cambridge Computing Service,
New Museums Site, Pembroke Street, Cambridge CB2 3QG, England.
Email: nmm1@cam.ac.uk
Tel.: +44 1223 334761 Fax: +44 1223 334679

Subject: Re: Bit counting and similar instructions
Date: Tue, 19 Jan 1999 18:40:34 -0800
From: alexr@I.HATE.SPAM (Alex Rosenberg)
Message-ID: <alexr-1901991840340001@roseal2.apple.com>
References: <36A47BD3.804DF5B0@hda.hydro.com>
Newsgroups: comp.arch.arithmetic
Lines: 50

In article <36A47BD3.804DF5B0@hda.hydro.com>, Terje Mathisen
<Terje.Mathisen@hda.hydro.com> wrote:

```
>Christian Bau wrote:
>> register. I think the Altivec processors (PowerPC 750 + Vector
>> instructions) could be quite fast at it as well. My guess is about 5
>> cycles for counting 128 bits.
>
>The Altivec might be even faster:
>
>The key would be to use the register-internal 16-element table lookup to
>convert 16 nibbles into the corresponding bit counts simultaneously.
>
>Doing this operation twice and adding should do the trick.
>
>I don't have an Altivec manual, but the (pseudo-)code should be
>something like this:
```

Motorola has a manual at:
<http://www.mot.com/SPS/PowerPC/teksupport/teklibrary/manuals/altivec_pem.pdf>

Apple also has a whole slew of AltiVec info at:
<http://developer.apple.com/hardware/altivec>
<http://developer.apple.com/tools/mpw-tools/altivec.html>

```
>  r1 = r0 >> 4;
>  r2 = 0x04030302030202010302020102010100;  // 16-element nibble table
>
>  r0 &= 0x0f0f0f0f0f0f0f0f0f0f0f0f0f0f0f0f;
>  r1 &= 0x0f0f0f0f0f0f0f0f0f0f0f0f0f0f0f0f;
>
>  r0 = register_table_lookup(r0,r2);
>  r1 = register_table_lookup(r1,r2);
>
>  return r0+r1;
```

The two masking operations are unnecessary. The permute operation is actually a 5-bit lookup and the 4-bit table can be specified twice to make the fifth bit indifferent. Also, this would only count within each byte. You'd have to add two sum across operations to compute all 128 bits. If you had a large number of bits to count, you could save a few cycles by accumulating a few iterations as bytes before the sum across operations.

Check artcle <6ja3v6$17p4$1@rtpnews.raleigh.ibm.com> by Brett Olsson
<brett@raleigh.ibm.com> in your favorite news archive for a 5-bit variant that's substantially similar.

```
+------------------------------------------------------------+
| Alexander M. Rosenberg           &LT;mailto:alexr@_spies.com> |
| Nobody cares what I say. Remove the underscore to mail me. |
```

Subject: Re: Bit counting and similar instructions
Date: 20 Jan 1999 18:44:51 -0800
From: gillies@cs.ubc.ca (Donald Gillies)
Message-ID: <7864b3keh1@cascade.cs.ubc.ca>
References: <alexr-1901991840340001@roseal2.apple.com>
Newsgroups: comp.arch.arithmetic
Lines: 47

It is not efficient to access memory to count bits on the PowerPC, or on ANY modern RISC processor. The cost of going to memory is roughly 5-10 instruction times. Even going to primary secondary cache is going to hurt your execution time a whole lot. Also, you cannot trust most RISC's to perform efficient byte accesses - many have to extract the byte slowly in the registers, or mask off the upper 24 bits, or whatnot. Moreover, many compilers (green hills) will throw away the base address of a lookup array after just 1 use, forcing a reload of this base address and wasting valuable cycles.

Thus, to count bits on most RISC's, make clever use of arithmetic:

```
#include "stdio.h"

// (C) Donald W. Gillies, 1992.  All rights reserved.  You may reuse
// this bitcount() function anywhere you please as long as you retain
// this Copyright Notice.
//
#define bitready()      register unsigned long tmp
#define bitcount(n)                                                     \
      (tmp = n - ((n >> 1) & 033333333333) - ((n >> 2) & 011111111111), \
      tmp = ((tmp + (tmp >> 3)) & 030707070707),                        \
      tmp =  (tmp + (tmp >> 6)),                                        \
      tmp = (tmp + (tmp >> 12) + (tmp >> 24)) & 077)

/* 16 instructions on most risc machines for 32-bit bitcount ! */

main()
{
        bitready();

        printf("bitcount(1) = %ld\n", bitcount(1));
        printf("bitcount(2) = %ld\n", bitcount(2));
        printf("bitcount(3) = %ld\n", bitcount(3));
        printf("bitcount(0xff) = %ld\n", bitcount(0xff));
        printf("bitcount(0xffffff) = %ld\n", bitcount(0xffffff));
}

% ~/a.out
bitcount(1) = 1
bitcount(2) = 1
bitcount(3) = 2
bitcount(0xff) = 8
bitcount(0xffffff) = 24
%
```

Subject: Re: Bit counting and similar instructions
Date: Thu, 21 Jan 1999 09:33:52 +0100
From: Terje Mathisen <Terje.Mathisen@hda.hydro.com>
Message-ID: <36A6E670.ECE65309@hda.hydro.com>
References: <7864b3keh1@cascade.cs.ubc.ca>
Newsgroups: comp.arch.arithmetic
Lines: 111

Donald Gillies wrote:

```
>
> It is not efficient to access memory to count bits on the PowerPC, or
> on ANY modern RISC processor.  The cost of going to memory is roughly
```

We were discussing two specific cpus here, both of which will actually run the table-lookup copy quickly, but in general, I agree.

```
> 5-10 instruction times.  Even going to primary secondary cache is
> going to hurt your execution time a whole lot.  Also, you cannot trust
> most RISC's to perform efficient byte accesses - many have to extract
> the byte slowly in the registers, or mask off the upper 24 bits, or
> whatnot.  Moreover, many compilers (green hills) will throw away the
> base address of a lookup array after just 1 use, forcing a reload of
> this base address and wasting valuable cycles.
```

This is a single example of a totally broken compiler, besides if the bitcount is actually time critical, I would implement it in asm anyway.

(I have actually written at leat 5 or 6 different asm versions of this code, see below)

```
> Thus, to count bits on most RISC's, make clever use of arithmetic:
>
> #include "stdio.h"
>
> // (C) Donald W. Gillies, 1992.  All rights reserved.  You may reuse
> // this bitcount() function anywhere you please as long as you retain
> // this Copyright Notice.
```

AFAIK, this algorithm was known before 1992.

```
> //
> #define bitready()      register unsigned long tmp
> #define bitcount(n)                                                     \
>       (tmp = n - ((n >> 1) & 033333333333) - ((n >> 2) & 011111111111), \
>       tmp = ((tmp + (tmp >> 3)) & 030707070707),                        \
>       tmp =  (tmp + (tmp >> 6)),                                        \
>       tmp = (tmp + (tmp >> 12) + (tmp >> 24)) & 077)
>
> /* 16 instructions on most risc machines for 32-bit bitcount ! */
```

To really make this version efficient, you need to do the bitcount across an array:

In that case you can speed it up a lot more, by the use of bit-parallel addition:

Three input words can be stored in two vertical words, with a single full adder to merge them.

Similarly seven input words can be converted to three vertical words, and finally you can convert 15 input words to 4 vertical words.

After doing the adds, the actual bitcounting in the resulting words should be handled in parallel, this gets rid of all/most of the stalls.

Robert Harley posted 64-bit Alpha code to do this which ran in (much?) less than a cycle/byte, I have written an MMX asm version which gets similar performance.

Here's an example for a seven-wide counter:

```
#define FULL_ADD(c1, c0, w0, w1, s1, s2) w1 = s1; c0 = w0; w0 &= w1; \
           c0 ^= w1; w1 = s2; c1 = c0; c0 ^= w1; c1 &= w1; c1 |= w0

#define MASK55 0x55555555
#define MASK33 0x33333333
#define MASK0F 0x0f0f0f0f

ulong count_bits7(unsigned *src)
{
    unsigned c0, c1, t0, t1, d0, d1, e0, e1, f1, f2;

    t0 = src[0];
    FULL_ADD(c1, c0, t0, t1, src[1], src[2]); // c1:c0         Up to 4
live vars + src[]
    FULL_ADD(d1, d0, c0, t1, src[3], src[4]); // d1:d0, c1     Up to 5
live vars + src[]
    FULL_ADD(e1, e0, d0, t1, src[5], src[6]); // e1:e0, d1, c1 Up to 6
live vars + src[]
    FULL_ADD(f2, f1, c1, t1,     d1,     e1); // f2:f1:e0

    e0 -= (e0 >> 1) & MASK55;                                   // 2 bits, 0-2
    f1 -= (f1 >> 1) & MASK55;
    f2 -= (f2 >> 1) & MASK55;

    e0 = (e0 & MASK33) + ((e0 >> 2) & MASK33);  // 4 bits, 0-4
    f1 = (f1 & MASK33) + ((f1 >> 2) & MASK33);
    f2 = (f2 & MASK33) + ((f2 >> 2) & MASK33);

    e0 += e0 >> 4; f1 += f1 >> 4; f2 += f2 >> 4; // 4 bits, 0-8
    e0 &= MASK0F; f1 &= MASK0F; f2 &= MASK0F;   // 8 bits, 0-8

    e0 += (f1 &LT;&LT; 1) + (f2 &LT;&LT; 2);            // 8 bits, 0-8+16+32 = 56
    e0 += e0 >> 8;                              // 8 bits, 0-112
    e0 += e0 >> 16;                             // 8 bits, 0-224

    return e0 & 255;
}
```

It is worth noting though that on a plain Pentium, which is very good at byte addressing, and can handle two loads/cycle, the naive byte-wide code ran neck&neck with this much more complicated algorithm!

Terje

--
- <Terje.Mathisen@hda.hydro.com>
Using self-discipline, see http://www.eiffel.com/discipline
"almost all programming can be viewed as an exercise in caching"

Terry Ritter, his current address, and his top page.

Last updated: 1999-02-20

Ritter's Latest Comments

From: ritter@io.com (Terry Ritter)
Newsgroups: sci.crypt,sci.math
Subject: Re: wanted: pascal source of strong RNG
Date: Fri, 12 Dec 1997 06:37:15 GMT
Lines: 35
Message-ID: <3490db92.9825304@news.io.com>
References: <6667n7bfl1@helga.keba.co.at> <348ED5B7.F6FE4A79@stud.uni-muenchen.de>
<348ee64e.21992473@nntp.ix.netcom.com> <snrvhww88iu.fsf@sable.ox.ac.uk>

On 11 Dec 1997 10:43:37 +0000, in <snrvhww88iu.fsf@sable.ox.ac.uk> in
sci.crypt Paul Leyland <pcl@sable.ox.ac.uk> wrote:

>[...]
>Also for most purposes, running a reasonably strong
>block cipher (such as or IDEA or Blowfish) in counter mode would be
>sufficient.

While I don't want to make too much of this, I would like to repeat my
long-standing position that using *any* real cipher or hash in
"counter mode" is disturbing and worrisome:

The problem is that a binary counter provides almost *ideal*
statistics for use in attacking a cipher or hash: the lsb always
changes; by selecting alternate states one can collect ciphertexts in
which one bit is in a known state; the next bit up changes at half the
rate, etc., etc. While this is of course no problem for the
conceptual *ideal* cipher or hash, the use of a binary counter in
practice demands more perfection from exactly those bit positions
which are least likely to provide it: those near the edge of the
computation.

There are alternatives: Use a polynomial counter, a fast preliminary
block cipher, or a randomizing mode like CBC. None of these have to
be secure (certainly a binary counter is not secure). The main thing
is to get about half of the bits changing about half of the time, so
The Opponent cannot exploit any practical statistical imperfections in
the first few bit positions of a real cipher or hash.

Terry Ritter ritter@io.com http://www.io.com/~ritter/
The Crypto Glossary: http://www.io.com/~ritter/GLOSSARY.HTM

From: ritter@io.com (Terry Ritter)
Newsgroups: sci.crypt
Subject: Re: ***Early RSA by CESG (1970)***
Date: Fri, 19 Dec 1997 23:22:45 GMT
Lines: 58
Message-ID: <349b0197.14341816@news.io.com>
References: <6781m8$m5h$1@pheidippides.axion.bt.co.uk> <67en9m$9kq$1@nntp.ucs.ubc.ca>

On 19 Dec 1997 21:00:38 GMT, in <67en9m$9kq$1@nntp.ucs.ubc.ca> in
sci.crypt unruh@physics.ubc.ca (Bill Unruh) wrote:

>Question about US patent law. I thought in the USA it is the first to
>invent, not the first to apply or the first to publish who gets the
>patent.

Yes.

>And demonstrating that something was invented by someone else
>earlier [...] breaks a patent. Is this true?

Yes.

>(even if kept secret)

Not as I see it. A patent is a reward for *publishing* an invention.
I think it is not uncommon for a patent to be awarded on an invention
that someone else was keeping a trade secret. The trade secret guy is
then out of luck, and the patent can be applied against him. (This
last is one of the things that the pending patent legislation was
intended to change, but I would expect that to lead to a vast increase
in court action and perjury as each company -- and maybe even the
military -- tried to show that it really had used each invention in
secret and so need not pay royalties.)

Basically, an inventor has the opportunity of availing himself of
either trade secret *or* patent protection, not both. As I
understand it, an application must be made in a reasonable time (say,
a year) unless necessarily delayed. One type of delay might be the
need to perform time-consuming experiments, and presumably there is
some sort of exception for bad legal advice, and perhaps even the lack
of funds or time. In general, though, the applicant must be seen as
"moving toward" a patent throughout the period before the application,
if the grant is contested. Someone not "moving toward" a patent
presumably has opted for trade secrecy protection (or, in this case,
military secrecy), and does not deserve a patent grant.

It is difficult to argue that one has benefited society in the sense
of the open exposure of new technology when one has kept that
technology secret.

>Do these UK
>inventions therefor break the RSA patent?

Not being a lawyer, I nevertheless doubt it.

Sorry.

Terry Ritter ritter@io.com http://www.io.com/~ritter/
The Crypto Glossary: http://www.io.com/~ritter/GLOSSARY.HTM

From: ritter@io.com (Terry Ritter)
Newsgroups: sci.crypt
Subject: Re: ***Early RSA by CESG (1970)***
Date: Mon, 22 Dec 1997 16:50:49 GMT
Lines: 43
Message-ID: <349e99f5.2855883@news.io.com>
References: <6781m8$m5h$1@pheidippides.axion.bt.co.uk> <67en9m$9kq$1@nntp.ucs.ubc.ca>
<67ldoh$h47$1@news.ox.ac.uk> <349E5A92.7C01@xtra.co.nz>

On Tue, 23 Dec 1997 00:18:26 +1200, in <349E5A92.7C01@xtra.co.nz> in
sci.crypt Peter Smith <lucent@xtra.co.nz> wrote:

>> > And demonstrating that something was invented by someone else
>> >earlier (even if kept secret) breaks a patent. Is this true? Do these UK
>> >inventions therefor break the RSA patent?
>>
>> I don't think that those are true -- and therefore, it doesn't break
>> the RSA patent.
>
>There is a patent principle known as 'prior art' which may vitiate this
>notion.

Alas, "prior art" is a "term of art," which is to say that it has a
particular meaning with respect patent law. This meaning differs from
the idea one might get from the ordinary meanings for the words
"prior" and "art."

In general, "prior art" in the patent sense refers to *open* or
known technology. This is mainly prior *publication*, especially
patent publication, but really any magazine article in a periodical
available in a library. Even formal ink-on-paper "publication" in a
company house-organ may *not* count as "prior art" if that was not
available to the public in libraries.

Secret development may have produced the same "art" and it may have
been "prior" to the patented invention and this may not be "prior art"
in the patent sense. The whole point of the patent system is to get
stuff openly *published* and widely known. Secret development does
not do this, and it is not counted the equal of development which
does.

In the normal case an inventor can choose trade secrecy *or* patent
protection, not both. It is not acceptable to keep a development
secret for years and then -- when the idea is re-invented -- cry:
"Prior art! Prior art!" That is not what "prior art" means.

Terry Ritter ritter@io.com http://www.io.com/~ritter/
The Crypto Glossary: http://www.io.com/~ritter/GLOSSARY.HTM

From: ritter@io.com (Terry Ritter)
Newsgroups: sci.crypt,talk.politics.crypto
Subject: Re: Q: Us Govt. standard(s) for classified data encryption
Date: Fri, 26 Dec 1997 18:34:23 GMT
Lines: 119
Message-ID: <34a3f86f.5301139@news.io.com>
References: <67hglb$5m7@sjx-ixn6.ix.netcom.com> <67fsrfqso1@sparky.wolfe.net>
<67hkom$8rr$1@news.ysu.edu> <349D4EC3.2B82@sprynet.com>
<wtshaw-2612970002240001@207.101.116.55> <wtshaw-2612971035030001@207.101.116.58>

On Fri, 26 Dec 1997 10:34:48 -0600, in
<wtshaw-2612971035030001@207.101.116.58> in sci.crypt
wtshaw@itexas.net (W T Shaw) wrote:

>[...]
>Someone wrote and asked how it would be proved that their were no
>backdoors. This would be proving a negative, but its merely a government
>request. It's par for the course for them to ask for the impossible from
>everyone so that they can make exception for anyone they want to win. On
>the other hand, clean engineering of an algorithm itself might make a
>backdoor pretty near impossible to place there.

The issue is not just "clean engineering," but also the *lack* of any
hidden structuring, such as one might find in particular tables.

This means that it is not enough to have a specific table which is
asserted to have been constructed with no hidden agenda, such as the
use of a particular irrational expansion. Many such constructions
could have been tried before one with the desired characteristics was
found. Consequently, it is necessary -- if there are tables, and one
wishes to avoid hidden structure -- to actually allow *random*
construction.

>Only complicated,
>verbose, inefficient design would allow for one in what should be a rather
>tight and plain area.

Obviously, any sort of "ad hoc" irregularity would be serious cause
for concern. But a cipher *could* be rather clever at disguising the
structure necessary to create a backdoor. By hiding it inside the
table construction, for example, there is no apparent part of the
design to be seen.

>Much todo was made about key analysis, however. The same might be said
>for finding bad keys. A keyspace might have to be relatively small so that
>a complete search for bad keys might be made. This is a non-starter since
>any keyspace of usable size tlo guarantee passable security needs to be
>huge so that it is hard to think about searching it.

A *scalable* cipher design can produce a large or small cipher from
the exact same construction rules. Just as we believe that
mathematics works the same for both small numbers and large numbers, a
backdoor cipher built from fixed construction rules must have the same
sort of backdoor, whether built large or small. With a scalable
cipher we can construct true small versions which *can* be
exhaustively examined. I have very recently reported on sci.crypt
results on the measured nonlinearity of 10-bit-wide Mixing
constructions; such measurements inherently require a scalable cipher
design.

>It is true that
>having a large keyspace might mean that hiding an bad key might make into
>some sort of back door. But, with the same size of selections in such a
>range, the odds of picking one at random should diminish.
>
>I return to memory of my question as to if key generation was considered
>part of the algorithm; government answer was yes. From my perspective,
>the answer is no. It is a wholely different thing. If the algorithm is
>good enough, its security depends solely on the keys, which should be
>variable as to equivalent bit strengths.

Well, certainly, if the cipher is known to *have* any substantial
quantity of weak keys, one would have to say that avoiding those keys
during key-generation *is* part of the cipher, so in this case "yes"
would be appropriate.

Ideally, of course, there would *be* no weak keys, and then it would
seem that "no" would be more appropriate. But since the keyspace had
to have been checked for this, one might say that this is what "yes"
was intended to convey.

The idea that we *can* generate efficient ciphers with really huge
keyspaces, so that there is absolutely no practical motive for using
keys of any particular restricted size, might be something the
government has not really understood or taken to heart. It may be
easier for the government to believe that restricting key size has
beneficial economic consequences for users, in addition to just
coincidentally supporting export limitations.

>While were at it, I'd sure like to debunk the idea that lack of randomness
>in output necessarily means that a poor algorithm was used, and
>consequently poor security is a result. This idea adopted by most, but it
>is a gross error to make it into a generalization as it can be
>specifically false. It is based on the idea that you can actually measure
>randomness, which you cannot really do.

I think you have just committed a logical error: You are arguing that
apparent randomness does not mean strength, which is true, and thus
that the lack of randomness does not mean weakness, which is false and
does not follow.

I claim that a cipher should indeed make plaintext (which inherently
has structure) into ciphertext with little or no structure.
Repeatable non-randomness in ciphertext would seem to be a clear
indication of a problem. One test might be to take a counting
sequence as plaintext, and then to try and find any sort of
relationship -- between bits or bytes or whatever -- in the
ciphertext.

>[...]
>Give me any so-called *random* ciphertext, and I can easily run it through
>a step to skew it somehow.

This is an interesting assertion, one perhaps best investigated
through actual examples. Personally, I suspect that the most obvious
ways of making randomness non-random would also destroy invertibility.

Terry Ritter ritter@io.com http://www.io.com/~ritter/
The Crypto Glossary: http://www.io.com/~ritter/GLOSSARY.HTM

From: ritter@io.com (Terry Ritter)
Newsgroups: sci.crypt
Subject: Re: Q: Us Govt. standard(s) for classified data encryption
Date: Sun, 28 Dec 1997 10:32:47 GMT
Lines: 104
Message-ID: <34a62a93.6255227@news.io.com>
References: <67hglb$5m7@sjx-ixn6.ix.netcom.com> <67fsrfqso1@sparky.wolfe.net>
<67hkom$8rr$1@news.ysu.edu> <349D4EC3.2B82@sprynet.com>
<wtshaw-2612970002240001@207.101.116.55> <wtshaw-2612971035030001@207.101.116.58>
<34a3f86f.5301139@news.io.com> <34A59D20.3D752504@ibm.net>

On Sat, 27 Dec 1997 19:28:16 -0500, in <34A59D20.3D752504@ibm.net> in
sci.crypt Uri Blumenthal <uri@ibm.net> wrote:

>Terry Ritter wrote:
>> A *scalable* cipher design can produce a large or small cipher from
>> the exact same construction rules. Just as we believe that
>> mathematics works the same for both small numbers and large numbers, a
>> backdoor cipher built from fixed construction rules must have the same
>> sort of backdoor, whether built large or small.
>
>I'm afraid I have to question the feasibility of the idea of a
>scalable cipher design. To illustrate what I mean, please
>consider (a) IDEA approach, and (b) what happens with an
>S-box when you increase its size...

Since you question feasibility, perhaps you will be moved to consider
my well-known and extensively-documented technology and designs:

http://www.io.com/~ritter/#MixTech

in particular:

http://www.io.com/~ritter/MIXCORE.HTM

and:

http://www.io.com/~ritter/#VSBCTech

in particular:

http://www.io.com/~ritter/VSBCCORE.HTM

With respect to (a), let me say that I personally have reservations
about the strength of IDEA. But this has nothing to do with scaling.
The reason IDEA does not scale up is the restrictive form of finite
field which is used in the design. I use fields of mod 2 polynomials,
and so avoid that type of scaling problem.

With respect to (b), it is trivially true that an S-box requires
exponential storage. But since all block ciphers are designed to
emulate a huge substitution table, it is quite clear that all real
block ciphers have design strategies intended to surmount this issue.

My designs generally fix the size of the tables, and then couple the
necessary number together with various forms of mixing; this supports
a dynamically-variable block size, either as a power-of-2 bytes, or an
arbitrary size to the byte. For exhaustive testing, however, it is
necessary to reduce the size of the tables so that the full codebook
can be traversed while still using multiple tables. See:

http://www.io.com/~ritter/ARTS/MIXNONLI.HTM

Clearly, any real block cipher must be qualitatively different from a
substitution table of the same size. However, this is an issue shared
by all block ciphers of realistic size, scalable or not.

>Possibly it [scalability] is too tough a nut to crack.

I encourage everyone to consider just how tough this is.

>On the
>bright side - I've yet to see the scalability problem addressed
>satisfactory in any network-related field (:-).

In other fields, scalability is generally related to throughput or
bandwidth. You will find this addressed in hardware realizations of
my Mixing designs:

http://www.io.com/~ritter/EXTRMSPD.HTM

In Mixing hardware, throughput can be increased arbitrarily by
increasing block size (along with transistor count), since the
hardware processing rate is constant *per* *block* no matter how large
that block may be. These ciphers are based on an arguably perfect
mixing of two bytes into two bytes; mixing in FFT-like patterns thus
mixes blocks of arbitrary power-of-2 size. For scalable throughput,
each mixer is implemented as separate hardware.

(Each factor-of-2 block size increase does require an additional
mixing sub-layer, which does increase latency somewhat, but this does
not affect throughput. For example, mixing 8 bytes requires 3 mixing
sub-layers, while mixing 64 bytes requires 6 mixing sub-layers; this
is twice the latency but eight times the throughput.)

In addition, the use of large blocks (e.g., 64 bytes or larger) may
support "electronic code book" (ECB) ciphering. This allows each
block to be processed separately, on separate hardware, which is
another form of throughput scalability.

So, yes, cipher scalability *can* scale throughput. But I see the
main scalability advantage being the ability to exhaustively examine
complete ciphering transformations (i.e., the complete codebook),
which is something simply not possible in normal designs.

Terry Ritter ritter@io.com http://www.io.com/~ritter/
The Crypto Glossary: http://www.io.com/~ritter/GLOSSARY.HTM

From: ritter@io.com (Terry Ritter)
Newsgroups: sci.crypt,talk.politics.crypto
Subject: Re: Q: Us Govt. standard(s) for classified data encryption
Date: Mon, 29 Dec 1997 17:55:39 GMT
Lines: 160
Message-ID: <34a7e3e9.2151126@news.io.com>
References: <67hglb$5m7@sjx-ixn6.ix.netcom.com> <67fsrfqso1@sparky.wolfe.net>
<67hkom$8rr$1@news.ysu.edu> <349D4EC3.2B82@sprynet.com>
<wtshaw-2612970002240001@207.101.116.55> <wtshaw-2612971035030001@207.101.116.58>
<34a3f86f.5301139@news.io.com> <wtshaw-2812970459590001@207.101.116.62>

On Sun, 28 Dec 1997 04:59:44 -0600, in
<wtshaw-2812970459590001@207.101.116.62> in sci.crypt
wtshaw@itexas.net (W T Shaw) wrote:

>In article <34a3f86f.5301139@news.io.com>, ritter@io.com (Terry Ritter) wrote:
>
>> On Fri, 26 Dec 1997 10:34:48 -0600, in
>> <wtshaw-2612971035030001@207.101.116.58> in sci.crypt
>> wtshaw@itexas.net (W T Shaw) wrote:
>>
>>[...]
>> This means that it is not enough to have a specific table which is
>> asserted to have been constructed with no hidden agenda, such as the
>> use of a particular irrational expansion. Many such constructions
>> could have been tried before one with the desired characteristics was
>> found. Consequently, it is necessary -- if there are tables, and one
>> wishes to avoid hidden structure -- to actually allow *random*
>> construction.
>
>Bad keys, bad tables, are bad keys.

But some cipher designs do claim to have table constructions above
reproach *because* they use a particular well-known irrational for
initialization.

>[...]
>What you say seems valid enough, that scaling-up is a good thing is worth
>doing, and simplifies testing.

Actually, my point is that a truly-scalable cipher allows scaling
down, which supports exhaustive testing that simply could not be
done on a larger cipher.

Scalability does far more than just *simplify* testing; it is an
enabling technology that supports experimental analysis which is
otherwise *impossible*.

>Weaknesses might be ingrained in scaling
>to certain sizes, sort of nodes dependent on the design if not flexible
>enough and/or the algorithm is based on some type of internal factor.

If it's not really scalable, it's not scalable.

>For instance, if an algorithm uses some sort of pRNG, it has to be tested
>at all levels intended as hidden loops in its calculations may create sort
>of an interference pattern with the specific parameters of the data
>allowed at that level. I've seen that before.

Personally, I have been able to accept a less perfectionist approach
which I think produces superior table initialization:

http://www.io.com/~ritter/KEYSHUF.HTM

I note that the output from that RNG is clean enough to expose a bias
in the runs-up / runs-down test, as I reported in sci.crypt:

http://www.io.com/~ritter/ARTS/RUNSUP.HTM

And while it is not possible to test for every potential correlation,
it *is* possible to *isolate* the generator so that even a highly
correlated sequence would seem to be difficult to attack.
Double-shuffling, for example, would appear to be a strong technique
for preventing knowledge of the table contents from disclosing the
shuffling sequence.

Correlations are possible in *any* cipher. Even though it is not
possible to test an RNG for every possible correlation, it is *also*
not possible to test every correlation (between bits or groups of
bits) in a block cipher. But we can hope to *isolate* the RNG
sequence from block ciphering operations, while the cipher itself
necessarily remains exposed.

>[...]
>I'm certainly open to error, but the point is one of correlation not
>proving causation, but it could demonstrate a close relationship if there
>was one. On the other hand, a high degree of correlation could be
>measured when samples are filtered in the first place for the desired
>end. Examples, even crypto examples, are easy to come by.

It is certainly possible to filter a mass of equivalent ciphertexts to
find a pattern you might want to present. But I expect this to be a
random process, requiring exponentially more ciphering for each
increase in desired pattern length.

This discussion was basically about the suggestion that even a cipher
which produces correlations could be OK. I assert again that this is
effectively always false. If a ciphering "core" (the cipher proper,
without massive filtering) does produce repeated correlations, and
this causes us to dismiss that cipher design, we will almost never be
wrong in so doing. So sayeth I.

>> I claim that a cipher should indeed make plaintext (which inherently
>> has structure) into ciphertext with little or no structure.
>> Repeatable non-randomness in ciphertext would seem to be a clear
>> indication of a problem. One test might be to take a counting
>> sequence as plaintext, and then to try and find any sort of
>> relationship -- between bits or bytes or whatever -- in the
>> ciphertext.
>
>Just what I would want you to do in some design, to look for things that
>were not there, and waste lots of time in the process.

Wouldn't it be easier, stronger, and faster just to use more keying or
more ciphering layers, or to encipher again under a complete other
cipher?

>> >[...]
>> >Give me any so-called *random* ciphertext, and I can easily run it through
>> >a step to skew it somehow.
>>
>> This is an interesting assertion, one perhaps best investigated
>> through actual examples. Personally, I suspect that the most obvious
>> ways of making randomness non-random would also destroy invertibility.
>>
>The cost of one process I have described before, phasing, is slight, a
>minimal increase in content, or a trade off of some sort, since getting
>something for nothing is not always obtainable. There are several obvious
>options for phasing protocols.

So, basically, we have more assertions but no examples.

If your claim is true, you ought to be able, in a paragraph or two, to
describe one such process in a way that most of us could understand
and agree that it does in fact do what you claim it does. Your
failure to explicitly do this when asked does not inspire confidence
in your claim.

>Having spent some more time today on transmutability of ciphertexts
>between various bases, even more possibilities for the sort of mischief I
>suggest become apparent.

And I would suspect that ciphertexts which are random-like in one base
are also random-like in another. Again, the more ciphertexts we have,
the more likely it is that we can select one which seemingly has
"patterns." But producing ciphertexts for pattern selection has a
cost which is exponential with expected pattern length, and the
expensive search and selection process buys only "mischief" with no
added real strength at all. Why would we do this?

>I intend to write a post to illustrate my quick
>method for discovering the more cryptopromising of the relationships
>between bases.
>[...]

Good.

Terry Ritter ritter@io.com http://www.io.com/~ritter/
The Crypto Glossary: http://www.io.com/~ritter/GLOSSARY.HTM

From: ritter@io.com (Terry Ritter)
Newsgroups: sci.crypt,talk.politics.crypto
Subject: Re: Q: Us Govt. standard(s) for classified data encryption
Date: Thu, 01 Jan 1998 20:29:38 GMT
Lines: 301
Message-ID: <34abfcac.9916179@news.io.com>
References: <67hglb$5m7@sjx-ixn6.ix.netcom.com> <67fsrfqso1@sparky.wolfe.net>
<67hkom$8rr$1@news.ysu.edu> <349D4EC3.2B82@sprynet.com>
<wtshaw-2612970002240001@207.101.116.55> <wtshaw-2612971035030001@207.101.116.58>
<34a3f86f.5301139@news.io.com> <wtshaw-2812970459590001@207.101.116.62>
<34a7e3e9.2151126@news.io.com> <wtshaw-2912971754400001@207.101.116.51>

Sorry for the delay; I've been trying to grok the essence . . . .

On Mon, 29 Dec 1997 17:54:24 -0600, in
<wtshaw-2912971754400001@207.101.116.51> in sci.crypt
wtshaw@itexas.net (W T Shaw) wrote:

>In article <34a7e3e9.2151126@news.io.com>, ritter@io.com (Terry Ritter) wrote:
>>
>> But some cipher designs do claim to have table constructions above
>> reproach *because* they use a particular well-known irrational for
>> initialization.
>
>If another could be substituted for it, then it is just an optional key
>choice. In the OTP marathon, the use of such a sequence as a pseudo came
>up. Even though it does not necessarily repeat, it is a known sequence.
>Whereever an irrational is used, might also a true random sequence be
>used, hense, more key choices? Supposedly, the sequences you mention are
>likely to not to be characterized as bad, but who knows.

The problem is that the definition of the cipher generally includes
table initialization, and using a particular irrational is *the*
defined way this is done in some ciphers.

Actually, I think this is related to your issue: Even though these
sequences might be considered "good" random-like sequences, they
obviously differ from each other. With an appropriate cipher design,
it might happen that one or another of these sequences would produce
an appropriate backdoor. Then it is just necessary to find a weak
one, and issue *that* irrational for use.

I think we can't trust tables that we cannot initialize at random,
even if their structure is claimed to be optimal.

>> >[...]
>> Scalability does far more than just *simplify* testing; it is an
>> enabling technology that supports experimental analysis which is
>> otherwise *impossible*.
>>
>Agreed.

Great!

>[...]
>Again, I would prefer to see that any key generation procedure passed the
>test of bad generation for any particular algorithm as in a particular
>implementation.

I have no idea what this means. How can we "pass" a test of "bad
generation"? What would such a test be?

>>[...]
>Having found one particular pRNG which worked generally, but had some
>problems in one implementation, I may be overcautious, but being burnt on
>occasion does that sort of thing to you. It is lots simpler to test when
>you do not know all the things you might test for but simply select a
>minimal few items, relying on chance to stumble on a problem area.

Building and testing RNG's is a time-honored sub-area of "computer
science" that is known to be non-trivial. People get burnt all the
time in this area. I did research and then a Cryptologia article on
the various possibilities, and the number of proposed schemes
continues to increase.

I suppose it helps to have more tests rather than fewer. But it seems
to me that isolating the RNG from any exposure greatly reduces any
risk that imperfections in the RNG can cause. It seems very important
to get an unbiased selection among tables, and a large-state RNG is
one way to do it. We *can* use other techniques, but then we have to
test *them* for essentially the same sort of problems, and that may be
much harder, because then we have to test the tables and the table
distribution directly. (Or just handwave it away, as is normally
done.) If we use the RNG technique, we can at least draw on the
experience that exists with those designs.

>>[...]
>> It is certainly possible to filter a mass of equivalent ciphertexts to
>> find a pattern you might want to present. But I expect this to be a
>> random process, requiring exponentially more ciphering for each
>> increase in desired pattern length.
>
>Not necessarily so, as certain algorithms rely on picking from a ready
>list of ciphertexts as the last step, just pick amongst the readily
>available strings.

. . . as in the last step of a cylinder cipher. This is the old
homophonic trick, right? But in a cylinder cipher the homophones are
related for a particular ciphering (selecting one also selects the
rest), where in most other techniques they are not. I suppose the
advantage is that one can work within the same alphabet instead of an
expanded alphabet, although the cost is an additional character --
beyond that needed for the plaintext -- which is used to indicate the
plaintext row.

But how much non-randomness can we expect, and can that amount
possibly be useful?

>> This discussion was basically about the suggestion that even a cipher
>> which produces correlations could be OK. I assert again that this is
>> *effectively* always false. If a ciphering "core" (the cipher proper,
>> without massive filtering) does produce repeated correlations, and
>> this causes us to dismiss that cipher design, we will almost never be
>> wrong in so doing. So sayeth I.
>
>If this is a simple result of a simplistic method, I would wholeheartedly
>concur, however, if you are dealing with a convoluted result, consciously
>skewed toward some predetermined non-random appearing distribution, having
>nothing to do with the underlying plaintext, any correlative method is
>screwed. I see this as not very difficult to cause.

OK, so the intent here is to select a pattern, a non-randomness,
independent of the ciphertext per se. For strength, we would seem to
want to select the pattern at random. So we are now introducing
non-randomness at random . . . .

The issue is whether this is practical. I'm still hoping to agree.

>>[...]
>> Wouldn't it be easier, stronger, and faster just to use more keying or
>> more ciphering layers, or to encipher again under a complete other
>> cipher?
>>
>The result would be obtained through added levels of key/algorithm,
>chained in a most productive example of good chaining. So, your
>suggestion is really my solution, in a way, but not producting random
>appearing results as a goal, but the contrary.

But, in this case, the ciphertexts do not all just appear, as in a
physical cylinder. In the computer we have to create them, before we
can select a pattern from among the random-like results. How can this
possibly be a reasonable strategy?

>[...]
>For my initial example, I'll use something similiar to that I have
>described before, making a 27 character based cipher look like a 24
>character based one:

I note in passing the we know that any such technique *must* expand
the ciphertext somehow. I assume that, here, the expansion will occur
in any case (the expansion necessary in an automatic cylinder cipher
to indicate which row holds the plaintext), and that this gives us the
opportunity to do something else with that expansion, which is
interesting.

I guess this leads me to question the amount of information introduced
by the row-indication character (or IV), since, if that were fully
utilized, there would be no freedom to do something else with it. Oh,
indeed, that *is* the case. We are foregoing a *random* row selection
to achieve our bias. And we had *better* get some good strength from
it, because we are giving up the "message key" properties of repeated
cylinder ciphering with random row selection.

>Initial Algorithm: Trinity 27
>PT: So, basically, we have more assertions but no examples.
>Keys from same text, just for demonstration purposes, normally you would
>not have them: Sub1: tpeaulfbm n/zgkcyho qridvwjsx
> Sub2: iqoudzypr g/antkhvw sebxjfclm
> Trans: /zdiayekn qfbstgrvj olucxwmph
>
>Preformatted text: Soj/basic allyj/we/ have/more /assertio ns/but/no
>/examples X/padding

I really don't understand the insertion of "/" and "j/"

>Groups:(7 blocks of 9 characters each)
>
>CT: fhvranphm goutwdmyx whhfdiaxp kwchnmiyq jzwitsdi/ ivxvcsrye qeqikafgw
>
>At this point, it seems that only one group has a "/" in it, but all will
>be treated alike,
>selectively encrypted with a cylinder. The protocol will use the first
>character in each group of ten, two groups of five, to indicate the
>original line. Of the possible 26 choices, any with a "/" , "O", or "A"
>in them will be rejected,

Of course we can go only so far with this. With a 26-char alphabet
and only 9 chars per block, we have a good chance of finding choices
which do not include 3 particular chars. But even here it seems to me
that there is no guarantee. We *might* not have such a row. Then
what?

>and a choice made of the remaining, with a "Q"
>in the strin. If, and this is the case, that if a "/" occurs in the
>reference position, which it will not this time, simply double the first
>letter in the CT2 group pair. The cylinder is 27 x 9.
>
>CT2: fjqwi jxsfe gjqwh qffuu wyvzg qphqm kwqtq rcyel jndnp qbiyx iqpbe
>qyfbi qjqhm zvwjf
>
>In the original message, there is no "F", no "G", and no "Q" and aside
>from punctuation, a rather typical alphabetic distibution.
>
>In CT, there is no "B" or "L", but there is a good distribution of letters.
>
>In CT2, lots of "Q" and lots of characters with a distribution on only one.

Well, I would say that any message has "a distribution" of characters.
It may be that only one has its normal distribution . . . .

>Larger passages where all letters were represented would still show that
>the selection process could assure that there could be a reduction in
>character(s),

That is, a reduction in the size of the ciphertext alphabet, not a
reduction in the number of characters in the message.

>and/or a selective peaking of the frequency of one or more,
>even at the same time.

Good, but it's expensive, and I worry about the strength. While we
don't have to produce multiple cipherings with a physical cylinder, in
the computer, we *do* have to produce those cipherings. So here we
are, doing 27 encryptions rather than just one, so that we can search
for various characteristics and so bias the ciphertext. This *is* the
expense. And of course we assume that The Opponents have the same
program we have, and so are aware of the possibility that we might do
this. And then the ciphertext alphabet tells them that we did. What
is strong about this?

>[...]
>> Again, the more ciphertexts we have,
>> the more likely it is that we can select one which seemingly has
>> "patterns."
>
>To build "patterns", like an extra lot of "Q's", you need do it only one
>point at a time, while to assimilate patterns, you try to work with as
>much as you can.

Yes, I see: For fairly limited goals, with small blocks, and the much
higher cost of making and ranking the multiple cipherings, bias *can*
be introduced. But we are not talking arbitrary patterns here. In a
sense, we are talking about re-codings, which have no strength at all.
We can *always* change the message coding, whether ciphering or not --
but the other end has to know it. We can make that part of the key,
if we have enough codings, but then The Opponents also know of that
possibility.

>> But producing ciphertexts for pattern selection has a
>> cost which is exponential with expected pattern length, and the
>> expensive search and selection process buys only "mischief" with no
>> added real strength at all. Why would we do this?
>
>The cost is only where you want it, with the cracker, not the encrypter
>where simple selection costs practically nothing. This is the argument
>against evolution, trying to make the whole seem impossible, like
>shuffling a box of loose clock parts and getting a working one from it,
>whereas, some careful stepwise selection in assembly means solving one
>obvious little problem at a time.

No, I think the cost is still in the enciphering operation. The cost
is just *hidden* on a physical cylinder, and so is easily discounted
in mental review. In practice, in a computer, there is a very
substantial cost involved (e.g., 1/26th the possible ciphering rate)
and a very significant worry about the strength of all this.

Yes, bias can be achieved. But we can do that by re-coding, which can
be a far more general and comprehensive solution, although I would not
expect that to add strength. We need it to add strength here.

>>
>An allusion to my notes can be in the list of probable bases for
>compatibility with base 27. It involves solving nth roots for values of
>powers of the selected base. Crude logs work OK for this. I just pulled
>them off of a scale on the trusty old slide rule, used some basics about
>these types of caluculations such as should be learned in JrHi, and
>confirmed the values on a calculator.
>
>An example is (27^4 = 531,441) < (14^5=537,824). So a suitable number of
>characters in base 27 will fit closely into a different number for base
>14, more since this is a situation of expansion in this conversion. The
>trick is to have *some*, but not overrun, head room in the new base since
>things are destined not to come out even in most cases. Hopefully, the
>disparity is in your favor.

It had *better* be in our favor, or arbitrary messages cannot be
transformed. This is another form of ciphertext expansion. We must
be able to represent at least as much information in the new base as
we have in the old one. The closeness of "at least" is the amount of
expansion.

But I am still searching for a handle on how much security all this
will provide in an automatic machine ciphering. How many of these

Hmm, interesting. Does not seem pick up a signature block in the
Thunderbird preview pane, but it does when I open the message.

Tony Walker - ICCondos.com - http://www.ic.com/ccnet/
The Public Forums - http://www.ic.com/ccnet/public/

From: ICCondos.com (Tony Walker)
Subject: Re: Is Bruce Schneier's Twofish cryptanalysis
Message-ID: <MPG.1d7d0b0b0b0b0b0b.ICCondos@news.ix.netcom.com>
Newsgroups: alt.cryptography,sci.crypt,comp.security.misc
Organization: ICCONDOS.COM(ICCONDOS@ix.netcom.com)
Lines: 3
Xref: news.netcom.com alt.cryptography:4 sci.crypt:1 comp.security.misc:1

In article <Twofish.BS@Counterpane.com>, ICCondos@ix.netcom.com (Tony Walker) wrote:

> Is Bruce Schneier's Twofish cryptanalysis a big deal?
> Or is it just a minor attack?
> - Tony Walker - ICCondos.com - http://www.ic.com/ccnet/

No.

From: ICCondos.com (Tony Walker)
Subject: Re: Is Bruce Schneier's Twofish cryptanalysis a big deal?
Message-ID: <MPG.1d7d0b0b0b0b0b0b.ICCondos@news.ix.netcom.com>
Newsgroups: alt.cryptography,sci.crypt,comp.security.misc
Organization: ICCONDOS.COM(ICCONDOS@ix.netcom.com)
Lines: 3
Xref: news.netcom.com alt.cryptography:5 sci.crypt:2 comp.security.misc:2

In article <Twofish.BS@Counterpane.com>, ICCondos@ix.netcom.com (Tony Walker) wrote:

> Is Bruce Schneier's Twofish cryptanalysis a big deal?
> Or is it just a minor attack?
> - Tony Walker - ICCondos.com - http://www.ic.com/ccnet/

No.

From: ICCondos.com (Tony Walker)
Subject: Re: Is Bruce Schneier's Twofish cryptanalysis a big deal?
Message-ID: <MPG.1d7d0b0b0b0b0b0b.ICCondos@news.ix.netcom.com>
Newsgroups: alt.cryptography,sci.crypt,comp.security.misc
Organization: ICCONDOS.COM(ICCONDOS@ix.netcom.com)
Lines: 3
Xref: news.netcom.com alt.cryptography:6 sci.crypt:3 comp.security.misc:3

In article <Twofish.BS@Counterpane.com>, ICCondos@ix.netcom.com (Tony Walker) wrote:

> Is Bruce Schneier's Twofish cryptanalysis a big deal?
> Or is it just a minor attack?
> - Tony Walker - ICCondos.com - http://www.ic.com/ccnet/

No.

From: ICCondos.com (Tony Walker)
Subject: Re: Is Bruce Schneier's Twofish cryptanalysis a big deal?
Message-ID: <MPG.1d7d0b0b0b0b0b0b.ICCondos@news.ix.netcom.com>
Newsgroups: alt.cryptography,sci.crypt,comp.security.misc
Organization: ICCONDOS.COM(ICCONDOS@ix.netcom.com)
Lines: 3
Xref: news.netcom.com alt.cryptography:7 sci.crypt:4 comp.security.misc:4

In article <Twofish.BS@Counterpane.com>, ICCondos@ix.netcom.com (Tony Walker) wrote:

> Is Bruce Schneier's Twofish cryptanalysis a big deal?
> Or is it just a minor attack?
> - Tony Walker - ICCondos.com - http://www.ic.com/ccnet/

No.

Tony Walker - ICCondos.com - http://www.ic.com/ccnet/
The Public Forums - http://www.ic.com/ccnet/public/

From: ICCondos.com (Tony Walker)
Subject: Re: Is Bruce Schneier's Twofish cryptanalysis a big deal?
Message-ID: <MPG.1d7d0b0b0b0b0b0b.ICCondos@news.ix.netcom.com>
Newsgroups: alt.cryptography,sci.crypt,comp.security.misc
Organization: ICCONDOS.COM(ICCONDOS@ix.netcom.com)
Lines: 3
Xref: news.netcom.com alt.cryptography:8 sci.crypt:5 comp.security.misc:5

In article <Twofish.BS@Counterpane.com>, ICCondos@ix.netcom.com (Tony Walker) wrote:

> Is Bruce Schneier's Twofish cryptanalysis a big deal?
> Or is it just a minor attack?
> - Tony Walker - ICCondos.com - http://www.ic.com/ccnet/

No.

Tony Walker - ICCondos.com - http://www.ic.com/ccnet/
The Public Forums - http://www.ic.com/ccnet/public/

From: ICCondos.com (Tony Walker)
Subject: Re: Is Bruce Schneier's Twofish cryptanalysis a big deal?
Message-ID: <MPG.1d7d0b0b0b0b0b0b.ICCondos@news.ix.netcom.com>
Newsgroups: alt.cryptography,sci.crypt,comp.security.misc
Organization: ICCONDOS.COM(ICCONDOS@ix.netcom.com)
Lines: 3
Xref: news.netcom.com alt.cryptography:9 sci.crypt:6 comp.security.misc:6

In article <Twofish.BS@Counterpane.com>, ICCondos@ix.netcom.com (Tony Walker) wrote:

> Is Bruce Schneier's Twofish cryptanalysis a big deal?
> Or is it just a minor attack?
> - Tony Walker - ICCondos.com - http://www.ic.com/ccnet/

No.

Tony Walker - ICCondos.com - http://www.ic.com/ccnet/
The Public Forums - http://www.ic.com/ccnet/public/

From: ICCondos.com (Tony Walker)
Subject: Re: Is Bruce Schneier's Twofish cryptanalysis a big deal?
Message-ID: <MPG.1d7d0b0b0b0b0b0b.ICCondos@news.ix.netcom.com>
Newsgroups: alt.cryptography,sci.crypt,comp.security.misc
Organization: ICCONDOS.COM(ICCONDOS@ix.netcom.com)
Lines: 3
Xref: news.netcom.com alt.cryptography:10 sci.crypt:7 comp.security.misc:7

In article <Twofish.BS@Counterpane.com>, ICCondos@ix.netcom.com (Tony Walker) wrote:

> Is Bruce Schneier's Twofish cryptanalysis a big deal?
> Or is it just a minor attack?
> - Tony Walker - ICCondos.com - http://www.ic.com/ccnet/

No.

Tony Walker - ICCondos.com - http://www.ic.com/ccnet/
The Public Forums - http://www.ic.com/ccnet/public/

From: ICCondos.com (Tony Walker)
Subject: Re: Is Bruce Schneier's Twofish cryptanalysis a big deal?
Message-ID: <MPG.1d7d0b0b0b0b0b0b.ICCondos@news.ix.netcom.com>
Newsgroups: alt.cryptography,sci.crypt,comp.security.misc
Organization: ICCONDOS.COM(ICCONDOS@ix.netcom.com)
Lines: 3
Xref: news.netcom.com alt.cryptography:11 sci.crypt:8 comp.security.misc:8

In article <Twofish.BS@Counterpane.com>, ICCondos@ix.netcom.com (Tony Walker) wrote:

> Is Bruce Schneier's Twofish cryptanalysis a big deal?
> Or is it just a minor attack?
> - Tony Walker - ICCondos.com - http://www.ic.com/ccnet/

No.

Tony Walker - ICCondos.com - http://www.ic.com/ccnet/
The Public Forums - http://www.ic.com/ccnet/public/

From: ICCondos.com (Tony Walker)
Subject: Re: Is Bruce Schneier's Twofish cryptanalysis a big deal?
Message-ID: <MPG.1d7d0b0b0b0b0b0b.ICCondos@news.ix.netcom.com>
Newsgroups: alt.cryptography,sci.crypt,comp.security.misc
Organization: ICCONDOS.COM(ICCONDOS@ix.netcom.com)
Lines: 3
Xref: news.netcom.com alt.cryptography:12 sci.crypt:9 comp.security.misc:9

In article <Twofish.BS@Counterpane.com>, ICCondos@ix.netcom.com (Tony Walker) wrote:

> Is Bruce Schneier's Twofish cryptanalysis a big deal?
> Or is it just a minor attack?
> - Tony Walker - ICCondos.com - http://www.ic.com/ccnet/

No.

Tony Walker - ICCondos.com - http://www.ic.com/ccnet/
The Public Forums - http://www.ic.com/ccnet/public/

From: ICCondos.com (Tony Walker)
Subject: Re: Is Bruce Schneier's Twofish cryptanalysis a big deal?
Message-ID: <MPG.1d7d0b0b0b0b0b0b.ICCondos@news.ix.netcom.com>
Newsgroups: alt.cryptography,sci.crypt,comp.security.misc
Organization: ICCONDOS.COM(ICCONDOS@ix.netcom.com)
Lines: 3
Xref: news.netcom.com alt.cryptography:13 sci.crypt:10 comp.security.misc:10

In article <Twofish.BS@Counterpane.com>, ICCondos@ix.netcom.com (Tony Walker) wrote:

> Is Bruce Schneier's Twofish cryptanalysis a big deal?
> Or is it just a minor attack?
> - Tony Walker - ICCondos.com - http://www.ic.com/ccnet/

No.

Tony Walker - ICCondos.com - http://www.ic.com/ccnet/
The Public Forums - http://www.ic.com/ccnet/public/

From: ICCondos.com (Tony Walker)
Subject: Re: Is Bruce Schneier's Twofish cryptanalysis a big deal?
Message-ID: <MPG.1d7d0b0b0b0b0b0b.ICCondos@news.ix.netcom.com>
Newsgroups: alt.cryptography,sci.crypt,comp.security.misc
Organization: ICCONDOS.COM(ICCONDOS@ix.netcom.com)
Lines: 3
Xref: news.netcom.com alt.cryptography:14 sci.crypt:11 comp.security.misc:11

In article <Twofish.BS@Counterpane.com>, ICCondos@ix.netcom.com (Tony Walker) wrote:

> Is Bruce Schneier's Twofish cryptanalysis a big deal?
> Or is it just a minor attack?
> - Tony Walker - ICCondos.com - http://www.ic.com/ccnet/

No.

Tony Walker - ICCondos.com - http://www.ic.com/ccnet/
The Public Forums - http://www.ic.com/ccnet/public/

From: ICCondos.com (Tony Walker)
Subject: Re: Is Bruce Schneier's Twofish cryptanalysis a big deal?
Message-ID: <MPG.1d7d0b0b0b0b0b0b.ICCondos@news.ix.netcom.com>
Newsgroups: alt.cryptography,sci.crypt,comp.security.misc
Organization: ICCONDOS.COM(ICCONDOS@ix.netcom.com)
Lines: 3
Xref: news.netcom.com alt.cryptography:15 sci.crypt:12 comp.security.misc:12

In article <Twofish.BS@Counterpane.com>, ICCondos@ix.netcom.com (Tony Walker) wrote:

> Is Bruce Schneier's Twofish cryptanalysis a big deal?
> Or is it just a minor attack?
> - Tony Walker - ICCondos.com - http://www.ic.com/ccnet/

No.

Tony Walker - ICCondos.com - http://www.ic.com/ccnet/
The Public Forums - http://www.ic.com/ccnet/public/

From: ICCondos.com (Tony Walker)
Subject: Re: Is Bruce Schneier's Twofish cryptanalysis a big deal?
Message-ID: <MPG.1d7d0b0b0b0b0b0b.ICCondos@news.ix.netcom.com>
Newsgroups: alt.cryptography,sci.crypt,comp.security.misc
Organization: ICCONDOS.COM(ICCONDOS@ix.netcom.com)
Lines: 3
Xref: news.netcom.com alt.cryptography:16 sci.crypt:13 comp.security.misc:13

In article <Twofish.BS@Counterpane.com>, ICCondos@ix.netcom.com (Tony Walker) wrote:

> Is Bruce Schneier's Twofish cryptanalysis a big deal?
> Or is it just a minor attack?
> - Tony Walker - ICCondos.com - http://www.ic.com/ccnet/

No.

Tony Walker - ICCondos.com - http://www.ic.com/ccnet/
The Public Forums - http://www.ic.com/ccnet/public/

Disc Mess, the unofficial parody, MPG-Express
Last updated: 1996-01-31

[Federal Register: February 16, 1996 (Volume 61, Number 33)]
[Rules and Regulations]
[Page 6111-6113]
From the Federal Register Online via GPO Access [wais.access.gpo.gov]

DEPARTMENT OF STATE
Bureau of Political Military Affairs

22 CFR Parts 123 and 126

[Public Notice 2294]

Amendment to the International Traffic in Arms Regulations

AGENCY: Department of State.

ACTION: Final rule.

SUMMARY: This rule would amend the International Traffic in Arms Regulations (ITAR) by establishing an exemption for the temporary export of cryptographic products for personal use. The effect of the change would be to ease the burden on U.S. citizens and lawful permanent residents who have the need to temporarily export cryptographic products when leaving the U.S. for brief periods of time.

EFFECTIVE DATE: February 16, 1996.

FOR FURTHER INFORMATION CONTACT:
Rose Biancaniello, Deputy Director for Licensing, Office of Defense Trade Controls, Department of State, (703) 875-6643 or FAX (703) 875-6647.

SUPPLEMENTARY INFORMATION: The U.S. Government has since 1993, at the direction of the President, been reviewing the U.S. policy regarding the domestic use of, and export controls on, cryptographic technology. While U.S. national security and foreign policy compel maintaining appropriate export controls on cryptography, the Department of State has continued to reform the export control procedures applicable to those products incorporating cryptography which are controlled by the ITAR in Category XIII(b)(1). For example, on September 2, 1994, the Department published (at 59 FR 45621) a final rule change which created a new Section 124.15. The section provides for a new arrangement by which the Department of State may provide approval for category XIII(b)(1) cryptography products to be distributed by U.S. manufacturers directly to foreign end users without obtaining an individual license for each transaction.

After extensive review, the Department of State has decided to further amend the regulations to provide for an exemption for the temporary export of cryptographic products for personal use. The exemption does not apply to other circumstances, for example, those in which a person contemplates sales, marketing or demonstration. Nor does the exemption apply to exports to destinations listed in Section 126.1 of the ITAR which are prohibited by a United Nations Security Council Resolution or to which the export (or for which the issuance of a

[[Page 6112]]
license for the export) would be prohibited by a U.S. statute (e.g., by Section 40 of the Arms Export Control Act, 22 U.S.C. 2780, to countries that have been determined to have repeatedly provided support for acts of international terrorism, i.e., Cuba, Iran, Iraq, Libya, North Korea, Sudan and Syria).

This rule amends Part 123 to add a new Section 123.27 to reduce the burden on individual users of cryptographic products by providing an exemption for the temporary export for personal use of products covered by Category XIII(b)(1) when the product remains in the possession of the exporter or the possession of another U.S. citizen or lawful permanent resident traveling with him/her. For purposes of this exemption, a product is considered to be in the possession of the exporter if the exporter takes normal precautions to ensure the security of the product by locking the product in a hotel room, safe, or other comparably secure location; and, while in transit, the exporter keeps the product in his/her carry-on luggage or locked in baggage accompanying the exporter which has been checked with the carrier.

This amendment involves a foreign affairs function of the United States and thus is excluded from the procedures of Executive Order 12866 (58 FR 51735) and 5 U.S.C. 553 and 554.

However, interested parties are invited to submit written comments to the Department of State, Director, Office of Defense Trade Controls, Attn: Regulatory Change, Personal Use Cryptographic Products, Room 200, SA-6, Washington, DC 20520-0602.

This rule affects collection of information subject to the Paperwork Reduction Act (44 U.S.C. 3501 et seq), and will serve to reduce the burden on exporters by adding an exemption which will remove the current requirement for a license.

Paperwork Reduction Act

The record-keeping requirement contained in section 123.27(b) has been approved by OMB and has a control number of 1405-0103. An agency may not conduct or sponsor, and a person is not required to respond to, a collection of information unless the collection of information displays a valid control number.

List of Subjects

22 CFR Part 123

Arms and munitions, Exports, Reporting and recordkeeping requirements.

22 CFR Part 126

Arms and munitions, Exports.

Accordingly, for the reasons set forth in the preamble, title 22, chapter I, subchapter M, of the Code of Federal Regulations, is amended as set forth below:

PART 123--LICENSES FOR THE EXPORT OF DEFENSE ARTICLES

1. The authority citation for part 123 continues to read as follows:

Authority: Secs. 2 and 38, Pub. L. 90-629, 90 Stat. 744 (22 U.S.C. 2752, 2778); E.O. 11958, 42 FR 4311, 3 CFR 1977 Comp. 79; 22 U.S.C. 2658.

2. A new Sec. 123.27 is added to read as follows:

Sec. 123.27 Temporary export for personal use of Category XIII(b)(1) cryptographic products.

(a) District Directors of Customs may permit a U.S. citizen or a U.S. person who is a lawful permanent resident as defined by 8 U.S.C. 1101(a)(20) to temporarily export from the United States without a license not more than one each of any unclassified Category XIII(b)(1) cryptographic hardware product and not more than a single copy of each type of unclassified Category XIII(b)(1) cryptographic software product provided that:

(1) The software product(s) are to be used only on a simultaneously temporarily exported Category XIII(b)(1) hardware product or a simultaneously exported item on the Commerce Control List (CCL); and

(2) The cryptographic products covered by Category XIII(b)(1) are not destined for export to a destination listed in Sec. 126.1 of the ITAR (22 CFR 126.1) which is prohibited by a United Nations Security Council Resolution or to which the export (or for which the issuance of a license for the export) would be prohibited by a U.S. statute (e.g., by Section 40 of the Arms Export Control Act, 22 U.S.C. 2780, to countries that have been determined to have repeatedly provided support for acts of international terrorism--currently Cuba, Iran, Iraq, Libya, North Korea, Sudan and Syria); and

(3)(i) The encryption products remain in the possession of the exporting person or the possession of another U.S. citizen or lawful permanent resident traveling with him/her, are for their exclusive use and not for copying, demonstration, marketing, sale, re-export or transfer of ownership or control. The export of cryptographic products identified in Category XIII(b)(1) in any other circumstances, for example, those in which a person contemplates sales, marketing, or demonstration must be licensed in accordance with policies and procedures established in this subchapter.

(ii) Special definition. For purposes of paragraph (a)(3)(i) of this section, a product is considered to be in the possession of the exporter if:

(A) The exporter takes normal precautions to ensure the security of the product by locking the product in a hotel room, safe, or other comparably secure location; and

(B) While in transit, the exporter keeps the product in his/her carry-on luggage or locked in baggage accompanying the exporter which has been checked with the carrier; and

(4) At the time of export from the U.S. and import into the U.S., the cryptographic products are with the individual's accompanying baggage or effects. They may not be exported or imported in unaccompanied baggage, mailed or transmitted by any other means (e.g., electronically); and, the cryptographic products must be returned to the U.S. at the completion of the stay abroad; and

(5) The exporter, upon request of a U.S. Customs officer, will submit the products to inspection at the time of export and/or import.

(b) Use of this exemption requires the exporter, in lieu of filing a Shippers' Export Declaration, to maintain, for a period of 5 years from the date of each temporary export, a record of that temporary export and the subsequent import. Included in this record must be a self certification that the individual complied with the conditions of paragraph (a) of this section and a self certification that he/she has no reason to believe that any of the temporarily exported cryptographic products were stolen, lost, copied, sold or otherwise compromised or transferred while abroad. The record should include the following information: A description of the unclassified cryptographic products; the countries entered, including the dates of entry and exit for each foreign country; and, the dates of temporary export from and subsequent import into the United States.

(c) In any instance where a product exported under this exemption is stolen, lost, copied, sold or otherwise compromised or transferred while abroad, the exporting person must, within 10 days of his/her return to the United States, report the incident to the Department of State, Office of Defense Trade Controls, Washington, D.C. 20520-0602. Also, any person who knows or has reason to know that cryptographic products exported under

[[Page 6113]]
this exemption are being transferred, exported, or used for any other activity which must be licensed or otherwise authorized in writing by the Department of State, should immediately inform the Department of State, Office of Defense Trade Controls, Washington D.C. 20520-0602.

PART 126--GENERAL POLICIES AND PROVISIONS

1. The authority citation for part 126 continues to read as follows:

Authority: Secs. 2, 38, 40, 42 and 71, Arms Export Control Act, Pub. L. 90-629, 90 Stat. 744 (22 U.S.C. 2752, 2778, 2780, 2791 and 2797); E.O. 11958, 41 FR 4311; E.O. 11322, 32 FR 119; 22 U.S.C. 2658; 22 U.S.C. 287c; E.O. 12918, 59 FR 28205.

2. Section 126.1(a) is amended by designating the three sentences of the undesignated paragraph as the third, fourth and fifth sentences of paragraph (a) and by adding a new sixth sentence at the end of paragraph (a) to read as follows:

Sec. 126.1 Prohibited exports and sales to certain countries.

(a) * * * With regard to Sec. 123.27 the exemption does not apply with respect to articles originating in or for export to countries prohibited by a United Nations Security Council Resolution or to which the export (or for which the issuance of a license for the export) would be prohibited by a U.S. statute (e.g., by Section 40 of the Arms Export Control Act, 22 U.S.C. 2780, to countries that have been determined to have repeatedly provided support for acts of international terrorism, i.e., Cuba, Iran, Iraq, Libya, North Korea, Sudan and Syria).
* * * * *

Dated: November 17, 1995.
Lynn E. Davis,
Under Secretary for Arms Control and International Security Affairs.
[FR Doc. 96-3190 Filed 2-15-96; 8:45 am]
BILLING CODE 4710-25-M

Source Code is Speech

Bernstein v. U.S. Department of State

Contents

- Brief Summary
- The Text
- Background
- Speech Analysis
- Source Code is Speech
- Cryptographic Source Code is Speech

Brief Summary

Bernstein designs a simple cipher and wishes to describe it and publish the C language source on the net. Believing the ITAR restrictions against export unjustly infringe his freedom of speech, Bernstein sues the government. In this preliminary action, Judge Patel of the United States District Court for the Northern District of California finds that this cryptographic source code is speech under the First Amendment. The actual case is decided months later.

The Text

From mingo@panix.com Thu Apr 18 22:23:59 CDT 1996
Article: 51496 of sci.crypt
Path:
news.io.com!insync!hunter.premier.net!news.mathworks.com!newsfeed.internetmci.com!panix!mingo
From: mingo@panix.com (Charlie Mingo)
Newsgroups: alt.security.pgp,sci.crypt
Subject: - 960415.decision (1/1) US Federal Court Rules Crypto is "Speech"
Date: Thu, 18 Apr 1996 23:01:46 -0400
Organization: 54 Barrow (via panix)
Lines: 881
Message-ID:
NNTP-Posting-Host: mingo.dialup.access.net
X-Newsreader: Yet Another NewsWatcher 2.2.0b7
Xref: news.io.com alt.security.pgp:56752 sci.crypt:51496

[NOTE: This file has some scanographical errors in it, but should be 99% legible.]

UNITED STATES DISTRICT COURT

NORTHERN DISTRICT OF CALIFORNIA

```
DANIEL J. BERNSTEIN,                |   No. C-95-0582 MHP
                                    |
               Plaintiff,           |   OPINION
                                    |
          vs.                       |
                                    |
UNITED STATES DEPARTMENT OF STATE   |
et al.,                             |
               Defendants.          |
____________________________________|
```

Plaintiff Daniel Bernstein brought this action against the Department of State and the individually named defendants seeking declaratory and injunctive relief from their enforcement of the Arms Export Control Act ("AECA"), 22 U.S.C. ¤ 2778, and the International Traffic in Arms Regulations ("ITAR"), 22 C.F.R. Sections 120-30 (1994), on the grounds that they are unconstitutional on their face and as applied to plaintiff. Now before this court is defendants' motion to dismiss for lack of justiciability.1

Having considered the parties' arguments and submissions, and for the reason set forth below, the court enters the following memorandum and order.

BACKGROUND 2

At the time this action was filed, plaintiff was a PhD candidate in mathematics at University of California at Berkeley working in the field of cryptography, an area of applied mathematics that seeks to develop confidentiality in electronic communication.

A. Cryptography

Encryption basically involves running a readable message known as "plaintext" through a computer program that translates the message according to an equation or algorithm into unreadable "ciphertext." Decryption is the translation back to plaintext when the message is received by someone with an appropriate "key." The message is both encrypted and decrypted by common keys. The uses of cryptography are far-ranging in an electronic age, from protecting personal messages over the Internet and transactions on bank ATMs to ensuring the of military intelligence.

As a graduate student, Bernstein developed an encryption algorithm he calls "Snuffle." He describes Snuffle as a zerodelay private-key encryption system. Complaint Exh. A. Bernstein has articulated his mathematical ideas in two ways: in an academic paper in English entitled "The Snuffle Encryption System," and in "source code" written in "C", a high-level computer programming language,3 detailing both the encryption and decryption, which he calls "Snuffle.c" and

"Unsnuffle.c", respectively. Once source code is converted into "object code," a binary system consisting of a series of 0s and 1s read by a computer, the computer is capable of encrypting and decrypting data. 4

B. Statutory and Regulatory Background

The Arms Export Control Act authorizes the President to control the import and export of defense articles and defense services by designating such items to the United States Munitions List ("USML"). 22 U.S.C. ¤ 2778(a)(1). Once on the USML, and unless otherwise exempted, a defense article or service requires a license before it can be imported or exported. 22 U.S.C. ¤ 2778(b)(2).

The International Traffic in Arms Regulations, 22 C.F.R. Sections 120-30, were promulgated by the Secretary of State, who was authorized by executive order to implement the AECA. The ITAR is administered primarily within the Department of State by the Director of the Office of Defense Trade Controls ("ODTC"), Bureau of Politico-Military Affairs. The ITAR allows for a "commodity jurisdiction procedure" by which the ODTC determines if an article or service is covered by the USML when doubt exists about an item. 22 C.F.R. ¤ 120.4(a).

Categories of items covered by the USNL are enumerated at section 121.1. Category XIII, Auxiliary Military Equipment, includes "Cryptographic (including key management) systems, equipment, assemblies, modules, integrated circuits, components

or software with the capability of maintaining secrecy or confidentiality of information or information systems .... __ _ ¤ 121 XIII(b)(l). A number of applications of cryptography are excluded, such as those used in automated teller machines and certain mass market software products that use encryption. Id.

C. Plaintiff's Commodity Jurisdiction Determinations On June 30, 1992 Bernstein submitted a commodity jurisdiction ("CJ") request to the State Department to determine whether three items were controlled by ITAR. Those items were Snuffle.c and Unsnuffle.c (together referred to as Snuffle 5.0), each submitted in C language source files, and his academic paper describing the Snuffle system. Complaint Exh. A. On August 20, 1992 the ODTC informed Bernstein that after consultation with the Departments of Commerce and Defense it had determined that the commodity Snuffle 5.0 was a defense article under Category XIII of the ITAR and subject to licensing by the Department of State prior to export. The ODTC identified the item as a "stand-alone cryptographic algorithm which is not incorporated into a finished software product." Complaint Exh. B. The ODTC further informed plaintiff that a commercial software product incorporating Snuffle 5.0 may not be subject to State Department control and should be submitted as a new commodity jurisdiction request.

Plaintiff and ODTC exchanged copious and contentious correspondence regarding the licensing requirements during the spring of 1993. Still unsure if his academic paper had been included in the ODTC CJ determination of August 20, 1992, Bernstein submitted a second CJ request on July 15, 1993, asking for a separate determination for each of five items. According to plaintiff these items were 1) the paper, "The Snuffle Encryption System," 2) Snuffle.c, 3) Unsnuffle.c, 4) a description in English of how to use Snuffle, and 5) instructions in English for programming a computer to use Snuffle.5 On October 5, 1993 the ODTC notified Bernstein that all of the referenced items were defense articles under Category XIII(b)(1). Complaint Exh. E; Defendant Exh. 18. After plaintiff initiated this action, the ODTC wrote to plaintiff to clarify that the CJ determinations pertained only to Snuffle.c and Unsnuffle.c and not to the three items of explanatory information, including the paper. Defendant Exh. 21. Bernstein appealed the first commodity jurisdiction determination on September 22, 1993. That appeal is still pending.

Plaintiff seeks to publish and communicate his ideas on cryptography. Because "export" under the ITAR includes "[d]isclosing . . . technical data to a foreign person, whether in the United States or abroad", Bernstein asserts that he is not free to teach the Snuffle algorithm, to disclose it at

academic conferences, or to publish it in journals or online discussion groups without a license.

LEGAL STANDARD

A motion to dismiss will be denied unless it appears that the plaintiff can prove no set of facts which would entitle him or her to relief. Conlev v. Gibson, 355 U.S. 41, 45-46 (1957); Fidelity Financial Corp. v. Federal Home Loan Bank of San Francisco, 792 F.2d 1432, 1435 (9th Cir. 1986), cert. denied, 479 U.S. 1064 (1987). All material allegations in the complaint will be taken as true and construed in the light most favorable to the plaintiff. NL Industries. Inc. v. Kaplan, 792 F.2d 896, 898 (9th Cir. 1986). Although the court is generally confined to consideration of the allegations in the pleadings, when the complaint is accompanied by attached documents, such documents are deemed part of the complaint and may be considered in evaluating the merits of a Rule 12(b)(6) motion. Durning v. First Boston Corp., 815 F.2d 1265, 1267 (9th Cir.), cert. denied sub. nom. Wyomina Community Dev. Auth. v. Durning, 484 U.S. 944 (1987).

DISCUSSION

Plaintiff makes a number of allegations of unconstitutionality with respect to the AECA and ITAR. Specifically, plaintiff alleges that the act and accompanying regulations, both facially and as applied, are a content-based

infringement on speech, act as an unconstitutional prior restraint on speech, are vague and overbroad, and infringe the rights of association and equal protection. Bernstein also alleges that the CJ request and registration processes as well as the licensing procedures are unconstitutional, although he does not state the basis of their unconstitutionality. Finally, plaintiff alleges that the actions of defendants are arbitrary and capricious and constitute an abuse of discretion under the Administrative Procedure Act, 5 U.S.C. Sections 701 et ~g Defendants move to dismiss on the grounds that these issues are nonjusticiable.

I. Justiciability The AECA plainly states:

The designation by the President (or by an official to whom the President's functions under subsection (a) of this section have been duly delegated), in regulations issued under this section, of items as defense articles or defense services for purposes of this section shall not be subject to judicial review. 22 U.S.C. ¤ 2778(h). Defendants conclude that this language, as well as the Constitution, precludes review of commodity jurisdiction determinations by this court. Plaintiff does not dispute this assessment. Defendants characterize this action as an attempt to obtain judicial review of their CJ determinations to place plaintiff's cryptographic items on the USML; as such, they maintain the action is precluded. However, this characterization does not comport with either the complaint

itself or plaintiff's repeated assertions that he is not seeking judicial review of defendants' CJ decision, but of the constitutionality of the statute and its regulations.

It is well established under the political question doctrine that courts do not have the expertise to examine sensitive political questions reserved for the other branches of government. See Baker v. Carr, 369 U.S. 186 (1962). More to the point, as defendants note, the determination of whether an item should be on the USML "possesses nearly every trait that the Supreme Court has enumerated traditionally renders a question 'political."' United States v. Martinez, 904 F.2d 601, 602 (llth Cir. 1990) (finding the CJ determination nonjusticiable without deciding if the then recent amendment to the AECA precluding judicial review applied to that case). However, a review of a particular CJ decision is a distinctly different question from a constitutional challenge to a statute. In Martinez, the Eleventh Circuit noted that defendants had not alleged a constitutional violation.6 904 F.2d at 603.

With respect to constitutional questions, the judicial branch not only possesses the requisite expertise to adjudicate these issues, it is also the best and final interpreter of them. Furthermore, as plaintiff points out, federal courts have consistently addressed constitutional issues in the context of national security concerns. See. e.q., New York Times Co. v. United States, 403 U.S. 713 (1971); Haia v. Aaee,

453 U.S. 280 (1981). Because the issues before this court do not necessitate a factual inquiry into the CJ determination, but a legal one into broader constitutional claims, the question is whether the statutory preclusion of judicial review of CJ decisions also embraces this court's review of the statute's constitutionality. 7

Defendants cite a number of Ninth Circuit cases that reject the reviewability of commodity designations under the analogous Export Administration Act, 50 U.S.C. App. Sections 2401 et seq., administered by the Commerce Department. Because this court is not reviewing the CJ determination itself, those cases miss the mark. Of those cases, however, United States v. Bozarov, 974 F.2d 1037 (9th Cir. 1992), cert. denied, 507 U.S. 917 (1993), is instructive.

In Bozarov the defendant was charged with exporting items on the Commerce Control List ("CCL")--which is akin to the USML--without a license in violation of the statute. The items, which were computer disk manufacturing equipment, had been listed on the CCL for national security reasons. Bozarov challenged the constitutionality of the Act's preclusion of judicial review. In upholding the preclusion of review, however, the court noted its decision was "bolstered by the fact that certain limited types of judicial review are available under the EAA despite the Act's seemingly absolute preclusion of review. First, colorable constitutional claims may be reviewed by the courts even when a statute otherwise

precludes judicial review." Id. at 1044 (citing Webster v. Doe, 486 U.S. 592, 602-05 (1988)). In fact, in order to reach the question of whether it was constitutional to preclude judicial review, the Ninth Circuit had to first find the issue justiciable. There, even the government conceded that Bozarov's nondelegation challenge amounted to a colorable constitutional claim. 974 F.2d at 1044 n.7. More definitive still is the Supreme Court's decision in Webster_, where it addressed whether employment decisions by the Director of the CIA were subject to judicial review. In Webster, plaintiff Doe was discharged from the CIA after informing the agency that he was a homosexual. He contested his termination partly on constitutional grounds. The Court held that the applicable statute bestowed so much discretion on the CIA Director in terminating employees that judicial review of those decisions was precluded under section 701(a)(2) of the APA. However, the Court made clear that such a holding did not preclude review of constitutional claims, noting that where Congress intends to preclude judicial review of constitutional claims its intent to do so must be clear.... We require this heightened showing in part to avoid the "serious constitutional question" that would arise if a federal statute were construed to deny any judicial forum for a colorable constitutional claim. 486 U.S. at 603 (citations omitted). 8 In the instant case, Congress has clearly precluded review of CJ determinations under the AECA, 22 U.S.C. ¤ 2778(h). But it has just as clearly tailored the preclusion of review to the designation by the President or his delegate "of items as

I defense articles or defense services for the purposes of this section." Id. Moreover, the language of section (h) indicates that it pertains only to delegations of the President's "functions under subsection (a) of this section." Those functions do not include constitutional determinations.

As this court finds that the AECA does not preclude judicial review of colorable constitutional claims, it must determine if plaintiff's claims are colorable in order to decide the issue of justiciability.

II. Colorability of Plaintiff's Constitutional Claims

Defendants maintain that plaintiff has raised no colorable constitutional claim because this case does not concern "speech" protected by the First Amendment, and even if it does, the minimal infringement is excusable under O'Brien v. United

U.S. 367 (1968). Defendant's further argue that plaintiff has not made a colorable claim that the CJ determinations constitute a prior restraint or that the AECA and ITAR are overbroad or vague.9 Plaintiff responds that the items that were subject to CJ determinations are speech of the most protected kind.

A. Analytical Framework

To determine if Bernstein states a "colorable constitutional claim," it is helpful to know what standard obtains. Colorability, a concept often employed by courts, is

rarely defined. Not surprisingly, discussions of colorability appear to be highly specific to both the claim and context in which they arise.

The Ninth Circuit has adopted the proposition that a constitutional claim is not colorable if it is clearly immaterial and made only for the purposes of jurisdiction, or "is wholly insubstantial or frivolous." Hoye v. Sullivan, 985 F.2d 990, 991-92 (9th Cir. 1993) (citing Boettcher v. Secretary of HHS, 759 F.2d 719, 722 (9th Cir. 1985)).

On a number of occasions the Ninth Circuit has addressed whether constitutional claims were colorable in the context of national security decisions. These have been largely due process and equal protection challenges to revocations of a security clearance. Dorfmont v. Brown, 913 F.2d 1399 (9th Cir. 1990), cert. denied, 499 U.S. 905 (1991); High Tech Gays v. Defense Ind. Sec. Clearance Off., 895 F.2d 563 (9th Cir.), reh'a denied, en banc, 909 F.2d 375 (1990); Dubbs v. CIA, 866 F.2d 1114 (9th Cir. 1989).

In Dorfmont the court held that there was no cognizable liberty or property interest in a security clearance that could give rise to a due process claim and therefore the claim was not colorable. The Dorfmont court noted, however, that it had found equal protection challenges to security clearance denials colorable in Hiah Tech Gays. 913 F.2d at 1403. In fact, in Hiah Tech Gays the court bypassed the issue of colorability altogether and concluded on the merits that homosexuals were

not a suspect or quasi-suspect class for purposes of heightened equal protection scrutiny.10 Plaintiffs in High Tech Gays had also brought a First Amendment claim based on freedom of association. The court found that plaintiffs had failed to allege or show a security clearance had been denied solely by reason of their membership in a gay organization and, therefore, there was no case or controversy with respect to that claim. In Dorfmont the court described its disposition of the First Amendment claim in Hiah Tech Gays as failure "to allege sufficient facts to raise a justiciable First Amendment claim." 913 F.2d at 1403 n.2. It is unclear whether the court's discussion of justiciability in Dorfmont applies to lack of colorabilty, and if so, what standard it implies. As Hoye is the most recently and clearly articulated of the Ninth Circuit's attempts to define colorability, its standard will govern the court's analysis in this case.

B. Analysis

Neither party agrees on exactly which items are at issue in this case, which confounds the analysis of whether subjecting them to a licensing requirement raises a colorable First Amendment claim. Defendants claim that only Snuffle.c and Unsnuffle.c are controlled by the USML and subject to the I licensing requirement. This is based on the 1995 letter the ODTC sent to plaintiff after he had filed suit in which it clarified that the CJ determinations did not include any

explanatory information, including the paper. This clarification would have been more appropriate in response to plaintiff's letter of July 15, 1993. Bernstein claims that his paper, "The Snuffle Encryption System," remains on the USML and that he has not been able to publish it without a license. It seems evident from the correspondence between Bernstein and the ODTC that the paper was indeed determined to be on the USML at the latest by October 5, 1993, but that as of June 29, 1995, the ODTC disavowed that decision. It is disquieting that an item defendants now contend could not be subject to regulation was apparently categorized as a defense article and subject to licensing for nearly two years, and was only reclassified after plaintiff initiated this action. Nonetheless, given defendants' reevaluation, the claims pertaining to the paper now appear moot.12

1. Speech

The paper, an academic writing explaining plaintiff's scientific work in the field of cryptography, is speech of the most protected kind. See Sweezv v. New Hampshire, 354 U.S. 234, 249-50 (1957) (noting the importance of protecting scholarship and academic inquiry). Nor do defendants contest this. Rather, defendants contend that Snuffle.c and Unsnuffle.c--the source code for the encryption program--are not speech but conduct. Plaintiff argues that computer code inscribed on paper, like any non-English language, is speech

protected by the First Amendment .13 Plaintiff further argues that even functional software is treated as protectable expression under copyright law. 14

Defendants urge this court to find the source code for Snuffle unprotected conduct rather than speech. They cite Texas v. Johnson, 491 U.S. 397 (1989), for the proposition that conduct must be "'sufficiently imbued with the elements of communication"' to fall within the protections of the First Amendment. Id. at 404 (quoting Spence v. Washington, 418 U.S. 405, 409 (1974)). In evaluating the communicative aspects of burning a flag in Texas v. Johnson, the Court framed the inquiry as whether the conduct entails an intent to convey a particular message and the likelihood of that message being understood. Id. According to defendants, the source code, as a functioning cryptographic product, is not intended to convey a particular message. It cannot be speech, they say, because its purpose is functional rather than communicative.

However, the Court in both Johnson and Spence, the flag desecration case upon which Johnson relies, inquired into the communicative nature of conduct only after concluding that the act at issue was indeed conduct and not speech. Both cases strongly imply that a court need only assess the expressiveness of conduct in the absence of "the spoken or written word." Johnson, 491 U.S. at 404; see Spence, 418 U.S. at 409 ("To be sure, appellant did not choose to articulate his views through printed or spoken words. It is therefore necessary to

determine whether his activity was sufficiently imbued with elements of communication to fall within the scope of the First and Fourteenth Amendments ...."). In the instant case, Bernstein's encryption system is written, albeit in computer language rather than in English. Furthermore, there is little about this functional writing to suggest it is more like conduct than speech. A computer program is so unlike flag burning and nude dancing that defendants' reliance on conduct cases is misplaced. It would be convoluted indeed to characterize Snuffle as conduct in order to determine how expressive it is when, at least formally, it appears to be speech. Recently the Ninth Circuit addressed the difference between speech and expressive conduct in assessing the constitutionality of the English-only provision amended to Arizona's constitution. Yniguez v. Arizonans for Official English, 69 F.3d 920, 934-36 (9th Cir. 1995) (en banc), cert. granted, 64 U.S.L.W. 3639 (U.S. Mar. 25, 1996) (No. 95-974). Defendants in Yniguez, like defendants here, sought to characterize one's choice of language as expressive conduct. The court was similarly "unpersuaded by the comparison between speaking languages other than English and burning flags." Id. at 934. The court further concluded that language was speech by definition:

> Of course, speech in any language consists of the ' expressive conduct' of vibrating one's vocal chords, moving one's mouth and thereby making sounds, or of putting pen to paper, or hand to keyboard. Yet the fact 16 that such 'conduct' is shaped by language--that is, a sophisticated and complex system of understood meanings--is what makes it speech. Language is by definition speech, and the regulation of any language is the regulation of speech.

Id. at 934-35. Nor does the particular language one chooses change the nature of language for First Amendment purposes. This court can find no meaningful difference between computer language, particularly high-level languages as defined above, and German or French. All participate in a complex system of understood meanings within specific communities. Even object code, which directly instructs the computer, operates as a "language." When the source code is converted into the object code "language," the object program still contains the text of the source program. The expression of ideas, commands, objectives and other contents of the source program are merely translated into machine-readable code. 15

Whether source code and object code are functional is immaterial to the analysis at this stage. Contrary to defendants' suggestion, the functionality of a language does not make it any less like speech. The Yniguez court noted that "the choice to use a given language may often simply be based on a pragmatic desire to convey information to someone so that they may understand it." Id. at 935. Thus, even if Snuffle source code, which is easily compiled into object code for the computer to read and easily used for encryption, is essentially functional, that does not remove it from the realm of speech. Instructions, do-it-yourself manuals, recipes, even technical information about hydrogen bomb construction, see United States v. The Progressive. Inc., 467 F. Supp. 990 (W.D. Wisc. 1979), are often purely functional; they are also speech.

Music, for example, is speech protected under the First Amendment. See Ward v. Rock Against Racism, 491 U.S. 781, 790 (1989). The music inscribed in code on the roll of a player piano is no less protected for being wholly functional. Like source code converted to object code, it "communicates" to and directs the instrument itself, rather than the musician, to produce the music. That does not mean it is not speech. Like music and mathematical equations, computer language is just that, language, and it communicates information either to a computer or to those who can read it. 16

Defendants argue in their reply that a description of software in English informs the intellect but source code actually allows someone to encrypt data. Defendants appear to insist that the higher the utility value of speech the less like speech it is. An extension of that argument assumes that once language allows one to actually do something, like play music or make lasagne, the language is no longer speech. The logic of this proposition is dubious at best. Its support in First Amendment law is nonexistent.

By analogy, copyright law also supports the "expressiveness" of computer programs. Computer software is subject to copyright protection as a "literary work." 17 U.S.C. Sections 101, 102(a)(1); accord Johnson Controls v. Phoenix Control Systems, 886 F.2d 1173, 1175 (9th Cir. 1989). For the purposes of copyright, literary works "are works, other than audiovisual works, expressed in words, numbers, or other verbal or numerical symbols or indicia, regardless of the nature of the material objects, such as books, periodicals, manuscripts, phonorecords, film, tapes, disks, or cards, in which they are embodied." 17 U.S.C. Section 101.

A computer program is further defined under the copyright statute as "a set of statements or instructions to be used directly or indirectly in a computer in order to bring about a certain result." Id. (emphasis added). Source code is essentially a set of instructions that is used indirectly in a computer since it must first be translated into object code to achieve the desired result. The statutory language, along with the caselaw of numerous circuits, supports the conclusion that copyright protection extends to both source code and object code. See NLFC. Inc. v. Devcom Mid-America. Inc., 45 F.3d 231, 234-35 (7th Cir.), cert. denied, 115 S.Ct. 2249 (1995) ("Both the source and object codes to computer software are also individually subject to copyright protection.") (citations omitted); Johnson Controls, 886 F.2d at 1175 ("Source code and object code, the literal components of a program, are consistently held protected by a copyright on the program.") (citations omitted); Apple Computer. Inc. v. Franklin Computer Corp., 714 F.2d 1240, 1249 (3d Cir. 1983), cert. dismissed, 464 U.S. 1033 (1984).

Copyright protection, designed to protect original expression, 17 U.S.C. Section 102(a), supports the likeness of a computer program to speech as defined by First Amendment law. The expression of an idea, a~ opposed to the idea itself, which is not afforded copyright protection under 17 U.S.C. Section 102(b), connotes the "speaking" of an idea. An encryption program expressed in source code communicates to other programmers and ultimately to the computer itself how to make the encryption algorithm (the idea) functional. Nor, under copyright law, does sheer functionality diminish the expressive quality of a copyrightable work. Apple Computer. Inc., 714 F.2d at 1252 (citing Mazer v. Stein, 347 U.S. 201, 218 (1954)); cf. Lotus Dev. Corp. v. Borland Int'l. Inc., 49 F.3d 807, 815 (lst Cir. 1995), judgment aff'd, 116 S.Ct. 804 (1996) (holding that a text describing how to operate something is subject to copyright protection while the method of operation itself is not). While copyright and First Amendment law are by no means coextensive, and the analogy between the two should not be stretched too far, copyright law does lend support to the conclusion that source code is a means of original expression.

For the purposes of First Amendment analysis, this court finds that **source code is speech**. Having concluded that all the items at issue, including **Snuffle.c and Unsnuffle.c are speech**, this court must now briefly review the claims defendants contest for colorability.

2. O'Brien

Defendants, relying on a characterization of Snuffle as conduct, argue that even if that conduct is expressive, the relatively mild O'Brien test should be employed. United States v. O'Brien, 391 U.S. 367 (1968), establishes the standard for assessing when a governmental regulation of conduct may nonetheless run afoul of the First Amendment's speech protections. Under O'Brien a regulation of conduct that incidentally restricts speech will be valid if 1) it is within the power of the government, 2) it furthers an important or substantial government interest, 3) the government interest is unrelated to the suppression of free expression and 4) the incidental restriction on speech is no greater than is essential to further that interest. Id. at 377.

Given that Snuffle source code is speech and not conduct, O'Brien does not appear to provide the appropriate standard under which to evaluate plaintiff's claims.17 However, as the parties have not had an opportunity to brief the issue of what First Amendment standard obtains, the court will apply O'Brien for the limited purpose of determining colorability. Defendants make a strong case that the AECA and ITAR satisfy the first and second prongs of O'Brien--that they are within the government's power and further the important interest of national security. With respect to prongs three and four, however, this court cannot say that plaintiff~s contentions are frivolous. Both the technical data provision of the ITAR, 22 C.F.R. Section 120.10, and Category XIII of the USML, 22 C.F.R. Section 121.1, regulating cryptographic software appear to relate to the "suppression of free expression" and may reach farther than is justifiable.

Defendants also argue that the Ninth Circuit's decision in United States v. Edler Industries Inc., 579 F.2d 516 (9th Cir. 1978), precludes a First Amendment attack under O'Brien on the AECA and its accompanying regulations. In Edler the court reviewed a conviction under the predecessor of the AECA for unlicensed exportation of technical data relating to a defense article on the USML. The technical data at issue in Edler related to a technique of tape wrapping with applications for missile components. After finding that "an expansive interpretation of technical data relating to items on the Munitions List could seriously impede scientific research and publishing and international scientific exchange," ~ at 519, the court went on to adopt a narrowing construction to save the statute. 18 Defendants urge that if Edler allows the government to legitimately restrict the export of technical data relating to a defense article, it can certainly restrict the defense article itself. Such an argument is an extension of Edler this court is unwilling to adopt. The validity of the of the munitions list was simply not at issue in that case. While Edler will be instructive to an analysis of the AECA under the First Amendment, it is sufficiently distinguishable on its facts that it cannot preclude plaintiff's challenge at this stage.

While the court makes no judgment on the merits, it finds plaintiff alleges facts sufficient to state a nonfrivolous First Amendment claim and hence that claim is colorable.

3. Prior Restraint

Plaintiff alleges that the AECA and ITAR act as an administrative licensing scheme for the publication of scientific papers, algorithms and computer programs related to cryptography, since publishing could release that information to foreign persons and would constitute exportation under the ITAR. 22 C.F.R. Section 120.17 .19

Governmental licensing schemes, such as the AECA and ITAR, come with a heavy presumption against their validity when they act as a prior restraint on speech. See Nebraska Press Assoc. v. Stuart, 427 U.S. 539 (1976); New York Times Co. v. United States, 403 U.S. 713 (1971) (per curiam); Near v. Minnesota, 283 U.S. 697 (1931). Prior restraints have even been struck down in the face of national security concerns. See e.a. New York Times, 403 U.S. at 714 (dissolving retraining order against newspaper publication of Pentagon Papers that included classified information). In New York Times the national security asserted was too vague a justification for prior restraints. Id. at 719 (Black, J., concurring), 725-26 (Brennan, J., concurring). In his concurrence to the per

curiam decision, Justice Stewart suggested a stringent test for permissible prior restraints, allowing them only when "disclosure . . . will surely result in direct, immediate, and irreparable damage to our Nation or its people." Id. at 730 (Stewart J., concurring). In response to the prior restraint claim, defendants rely on the argument rejected above, that Snuffle is not speech and does not implicate the First Amendment.

Since Snuffle is speech that is potentially subject to the prior restraint of licensing, and under the AECA that restraint is unreviewable, plaintiff's prior restraint claim is colorable. 20

4. Overbreadth

Plaintiff alleges that the AECA and ITAR are overbroad with respect to their regulation of items with predominately civil applications, the definition of export, Category XIII of the USNL, and the definition of software.

Defendants rely extensively on Edler to argue that any overbreadth challenge is foreclosed to plaintiff because the Ninth Circuit has provided a limiting construction to the technical data provision. They also cite the 1984 revisions to ITAR which they contend are even more solicitous of speech because they provide for certain exemptions from technical data for academic research and information in the "public domain." Defendant Exh. lA. However, plaintiff's overbreadth claim goes

beyond the technical data provision and beyond those items classified as technical data. The complaint makes clear that the challenge is significantly broader than the scope of Edler and pertains to the defense articles themselves.

Facial overbreadth is concededly "strong medicine" employed as a last resort when a limiting construction cannot be applied to a statute. Broadrick v. Oklahoma, 413 U.S. 601, 613 (1973). Defendants employ Broadrick to propose that when conduct as well as speech is regulated, the overbreadth must be substantial in relation to the statute's legitimate sweep. Id. at 615. However, in a subsequent Supreme Court decision relied upon by defendants, Members of the City Council of Los Angeles v. Taxpayers for Vincent, 466 U.S. 789 (1984), the Court noted that "where the statute unquestionably attaches sanctions to protected conduct, the likelihood that the statute will deter that conduct is ordinarily sufficiently great to justify an overbreadth attack." Id. at 801 n.l9 (citing Erznoznik v. City of Jacksonville, 422 U.S. 205 (1975)). In Taxpayers for Vincent the Court clarified the application of substantial facial overbreadth, saying there must be a "realistic danger that the statute itself will significantly compromise recognized First Amendment protections of parties not before the Court ...." Id. at 801. Merely being able to conceive of "some impermissible applications of a statute" is insufficient. Id. at 800.

As this court has noted above, **cryptographic source code is speech**. Even if the statute aims at conduct as well as speech so as to invoke the "substantial overbreadth" doctrine, the court at this stage of the proceedings need only determine whether the claim is colorable. On the record before it at this time, the court cannot say that plaintiff's claim that enforcement of some provisions of the statute or regulations could significantly compromise the protected speech of third parties is frivolous.

5. Vagueness

Plaintiff alleges that a number of terms and provisions within the AECA and ITAR are impermissibly vague in that they fail to give notice of the conduct they regulate and have a chilling effect on speech. These provisions include inter alia the meaning of software capable of maintaining secrecy under Category XIII of the USML, the exemptions for information taught in universities, the definition of public domain, and the "willful" requirement for criminal penalties.

For a claim of facial vagueness to survive, the deterrent effect of the statute on protected expression must be "real and substantial" and not easily narrowed by a court. Young v. American Mini Theaters. Inc., 427 U.S. 50, 60 (1976). Defendants again rely heavily on Edler to argue that the Ninth Circuit has already resolved the problems plaintiff challenges. While this may be true of the technical data provision, it

leaves unaddressed numerous other areas of concern. Defendants also conclude summarily that both the definition of cryptographic software and the exemptions from this definition are clear to a person of ordinary intelligence. This seems to be a bit of dissimulation, unless it is a confession, since the ODTC itself mistakenly classified Bernstein's academic paper as a defense article under Category XIII. Finally, defendants contest plaintiff's vagueness challenge to the "willful" requirement for criminal penalties, citing the Ninth Circuit's clarification that under the AECA willfulness requires a "voluntary, intentional violation of known legal duty ...." United States v. Lizarraga-Lizarraga, 541 F.2d 826, 828 (1976) (construing the predecessor to the AECA). According to Posters 'N' Things. Ltd. v. United States, _ U.S. _ , 114 S. Ct. 1747, 1754 (1994), such a scienter requirement helps to avoid the problem of vagueness a criminal statute might otherwise allow.

With the exception of the claim against the willful standard for criminal violations of the AECA, this court does not find plaintiff's claims of vagueness frivolous.

It should be emphasized that with the exception of its conclusions that source code is speech for the purposes of the First Amendment and that this case is justiciable, the court makes no other substantive holdings.

CONCLUSION

For the reasons set forth above, IT IS HEREBY ORDERED that defendants' motion to dismiss is DENIED.

IT IS SO ORDERED.

Dated: April 15, 1996 MARILYN HALL PATEL
United States District Judge

ENDNOTES

1. Defendants pose the justiciability issue as one of subject matter jurisdiction. As those questions are distinct and defendants arguments go to justiciability, this court addresses the motion as one pertaining to justiciability alone. See Baker v. Carr, 369 U.S. 186, 198 (1962).

2. Except where noted, these facts come from undisputed portions of the record.

3. Source code is the text of a source program and is generally written in a high-level language that is two or more steps removed from machine language which is a low-level language. High-level languages are closer to natural language than lowlevel languages which direct the functioning of the computer. Source code must be translated by way of a translating program into machine language before it can be read by a computer. The object code is the output of that translation. It is possible to write a source program in high-level language without knowing about the actual functions of the computer that carry out the program. Encyclopedia of Computer Science 962, 1263-64 (Anthony Ralston & Edwin D. Reilly eds., 3d ed. 1995)

4. The parties disagree about whether the computer code submitted by plaintiff to the State Department is technically "software." Defendants refer to the computer code as software even though it i8 not in object code on a disk. Plaintiff contests this characterization. In any event, in order to be software, which are instructions to the computer, the instructions must be in a form that can be easily altered as distinguished from firmware or hardware which cannot be readily altered, if it can be altered at all.

The court notes that 22 CFR Section 121.8(f) defines "software" for the purposes of the AECA. That definition is descriptive of content, however, and does not define the actual format or physical form of the software. At this stage the court need not resolve this issue since whatever the program's form, the ODTC has subjected it to the licensing requirements.

5. The CJ request of July 15, 1993, refers to the items as W BCJF-2, W BCJF-3, DIBCJF-4, DJBCJF-5, and W BCJF-6 without distinguishing information. Complaint Exh. D.

6. This statement appears to be contradicted by that court's own reference to defendants' overbreadth claim on the preceding page of its opinion. Martinez, 904 F.2d at 601. It is not clear whether the overbreadth argument went to constitutionality or merely to statutory interpretation.

7. Plaintiff argues that this court has power to review his cause of action under a political question analysis. Even if that were so, he fails to consider the effect of a clear statement by Congress precluding judicial review in the context of the AECA. Furthermore, plaintiff dedicates nearly ten pages of his brief in opposition to this motion to arguing that review is proper under the Administrative Procedure Act ("APA"). However, as defendants note, to the extent judicial review is precluded by statute, it is also precluded by the APA. 5 U.S.C. Section 701(a)(1) ("This chapter applies . . . except to the extent that--(l) statutes preclude judicial review ...."). That does not necessarily mean plaintiff's allegation that defendants exceeded their lawful authority under the APA is unreviewable. Plaintiff is correct that U.S. v. Bozarov allows courts to exercise review, in the face of statutory preclusion, of "claims that the Secretary acted in excess of his delegated authority under the EAA." 974 F.2d at 1045. Nonetheless, defendants only argue nonjusticiability based on the First Amendment claim. This court declines to rule on the colorability of every one of plaintiff's claims without briefing on those issues. Currently before the court is simply the issue of the justiciability of plaintiff's First Amendment challenge.

8. The Court did not consider whether Doe presented a colorable constitutional claim because that question was not properly before the Court.

9. Defendants only argue in passing that plaintiff's claim that the CJ determinations were made in excess of statutory authority is not justiciable.

10. The discussion of Hiah Tech Gays in Dorfmont betrays the unusual procedural posture the Ninth Circuit adopted in order to reach the merits: "Without addressing whether the federal courts have jurisdiction to hear these claims, we ruled in favor of defendants on the merits of the equal protection attack." 913 F.2d at 1403 (emphasis added) (citation omitted).

11. Reading "colorable" to mean sufficient to state a claim, or even nonfrivolous, is supported by the Sixth Circuit's decision in Brooks v. Seiter, 7-79 F.2d 1177, 1181 (6th Cir. 1985), in which the court, using a frivolousness standard, held that plaintiff prisoners had alleged a First Amendment violation when they complained that prison officials withheld mail order publications. In the context of that holding, the court said that the state interest in deferring to prison officials did not bar courts from hearing a "colorable constitutional claim." Id.

12. If there is any uncertainty about this, defendants should state their determination without equivocation so that the mootness issue can be completely resolved as soon as possible.

13. Bernstein also contends that encryption software is important not only as speech, but as a tool to protect private speech. Plaintiff argues that cases protecting anonymous speech and prohibiting compelled speech support this novel proposition. However, certainly at this stage, the court need not reach the issue.

14. Plaintiff briefly argues that his encryption program, written in source code on paper, is not functional at all. Given the ease with which one can convert source code into object code, however, this argument is specious. More to the point is plaintiff's contention that source code and functioning software are both fully protected under the First Amendment.

15. The court does not employ the word "translate" as art thereby excluding the applicability of "compile", "interpret" or related terms.

16. Whether such "languages" as assembly language or low-level languages constitute speech, or may sometimes constitute speech, need not be addressed at this time in view of the court's ruling that the source code provides the basis for a colorable claim.

17. Plaintiff cites Justice Department memoranda that question the constitutionality of some of the ITAR provisions as well as the propriety of an O'Brien analysis. Plaintiff Exh. A at 60007, 60090. A 1978 memo from the DOJ Office of Legal Counsel addressing the constitutionality of the ITAR restrictions on public cryptography noted that "even a cursory reading of the technical data provisions reveals

Some large portions of the FCAM are devoted to summarization. A more
detailed discussion of the topics covered in the FCAM can be found in the
accompany "Principles" or in Volume I. In this section, we concentrate
on the aspects of the FCAM that are specific to the Pentagon and
drive directly regulatory policy-originating.

The FCAM is a document that ties together information from the studies and
analyses of a variety of sources and presents a unified view "Significant"
issues are those that have a significant impact on the budget, whether
it be up or down.

The FCAM is a document that brings these principles and disciplines about the
all-volume of the academic paper. In its presentation-mode mode it has
the same "Executive Summary" and "Introduction" sections that a
paper and have particular to be a defense-force-wide mathematical
model. Unlike a paper, it has a section on "Data" and "Assumptions"
that has been specific to the model. It also includes a
"Methodology" section that describes the model and the
principles of the policy. Where appropriate, it also has sub-sections
on "Sensitivity" and "Uncertainty" that are specific to the model of this
year.

The Pentagonese and particular work related to the has controversial
issues particular to the defense-mathematical-model environment.
Some particular issues are the "Sensitivity" and "Uncertainty" issue seems
to be particular to the model. The "Sensitivity" section is
implicitly to be particular to the model because the financial
"Sensitivity" section is of a "Uncertainty" or "Uncertainty" or "Uncertainty"
nature.

Generated by Unknown HTML to Markdown v0.7.0

A Crypto Algorithm for the Twenty-first Century . . .

Advanced Encryption Standard (AES) Development Effort

This page can now be reached via **http://www.nist.gov/aes**

NIST HOME

GENERAL

- AES Home Page
- Recent News

ROUND 2 (8/1999-5/2000)

- AES Finalists: NIST's Round 1 Report
- Finalist Algorithm Information
- R2 Analysis
- Submit Comments
- R2 Comments
- Discussion Forum
- AES CD-ROMs
- 3rd AES Conference

ROUND 1 (8/1998-4/1999)

- R1 Algorithms
- R1 Announcement
- R1 Comments
- 2nd AES Conference
- 1st AES Conference

Pre-ROUND 1 (1/1997-7/1998)

- Call for Candidates
- AES Beginnings

Recent Announcements

March 21-22, 2000 - Papers submitted for AES3 are now available, along with a preliminary agenda for AES3. The finalist algorithm submitters have provided NIST with updates to their original intellectual property statements.

February 18, 2000 - **AES Finalist Source Code is now available**, which means that all of the finalist algorithm information found on AES CD-3 is now available on these AES home pages.

December 8, 1999 - Beginning today, NIST will make available Round 2 public comments received-to-date. This shall be updated every other Wednesday for the remainder of Round 2.

November 1, 1999 - NIST has developed a short white paper [PDF] to promote and focus the discussion of important Round 2 issues.

Additional AES "News" during Round 2.

August 9, 1999 - **NIST Announces the AES Finalist Candidates for Round 2:**

MARS, RC6™, Rijndael,Serpent, and Twofish

A press release is available for this announcement; a bulletin from NIST's Informtation Technology Laboratory (ITL) gives a status report on the AES (August 1999)

NIST's Round 1 Report is available which summarizes the results of Round 1 analysis, and describes the selection of the AES finalists. (This report is scheduled to be published in the NIST Journal of Research.)

(Addenda [PDF] for the Round 1 Report are now available.)

There were four modification ("tweak") proposals received from submitters after Round 1. The modification proposed for MARS was accepted.

Official public comments on the Round 2 finalists may be submitted electronically to AESround2@nist.gov. (A Federal Register announcement will be forthcoming in the near future.)

SPECIAL NOTE - Intellectual Property

NIST reminds all interested parties that the adoption of AES is being conducted as an open standards-setting activity. Specifically, NIST has requested that all interested parties identify to NIST any patents or inventions that may be required for the use of AES. NIST hereby gives public notice that it may seek redress under the antitrust laws of the United States against any party in the future who might seek to exercise patent rights against any user of AES that have not been disclosed to NIST in response to this request for information.

Overview of the AES Development Effort

August, 1999

The National Institute of Standards and Technology (NIST) has been working with industry and the cryptographic community to develop an Advanced Encryption Standard (AES). The overall goal is to develop a Federal Information Processing Standard (FIPS) that specifies an encryption algorithm(s) capable of protecting sensitive government information well into the next century. The algorithm(s) is expected to be used by the U.S. Government and, on a voluntary basis, by the private sector.

On January 2, 1997, NIST announced the initiation of the AES development effort and made a formal call for algorithms on September 12, 1997. The call stipulated that the AES would specify an unclassified, publicly disclosed encryption algorithm(s), available royalty-free, worldwide. In addition, the algorithm(s) must implement symmetric key cryptography as a block cipher and (at a minimum) support block sizes of 128-bits and key sizes of 128-, 192-, and 256-bits.

On August 20, 1998, NIST announced a group of fifteen AES candidate algorithms at the First AES Candidate Conference (AES1). These algorithms had been submitted by members of the cryptographic community from around the world. At that conference and in a simultaneously published Federal Register notice, NIST solicited public comments on the candidates. A Second AES Candidate Conference (AES2) was held in March 1999 to discuss the results of the analysis conducted by the global cryptographic community on the candidate algorithms. The public comment period on the initial review of the algorithms closed on April 15, 1999. Using the analyses and comments received, NIST selected five algorithms from the fifteen.

The AES finalist candidate algorithms are **MARS, RC6, Rijndael, Serpent, and Twofish**. NIST has developed a Round 1 Report describing the selection of the finalists.

These finalist algorithms will receive further analysis during a second, more in-depth review period prior to the selection of the final algorithm(s) for the AES FIPS. NIST solicits comments on the remaining algorithms through May 15, 2000. Comments and analysis are actively sought by NIST on any aspect of the candidate algorithms, including, - but not limited to, - the following topics: cryptanalysis, intellectual property, crosscutting analyses of all of the AES finalists, overall recommendations and implementation issues. An informal AES discussion forum is also provided by NIST for interested parties to discuss the AES finalists and relevant AES issues.

Near the end of Round 2, NIST will sponsor the Third AES Candidate Conference (AES3) - an open, public forum for discussion of the analyses of the AES finalists. Submitters of the AES finalists will be invited to attend and engage in discussions regarding comments on their algorithms. AES3 will be held April 13-14, 2000 in New York, NY, USA. Proposed papers for this conference are due to NIST by January 15, 2000, and they will also be considered as Round 2 public comments.

Following the close of the Round 2 public analysis period on May 15, 2000, NIST intends to study all available information and propose the AES, which will incorporate one or more AES algorithms selected from the finalists. The AES will be announced as a proposed Federal Information Processing Standard (FIPS) which will be published for public review and comment. Following the comment period, the standard will be revised, as appropriate, by NIST in response to those comments. A review, approval, and promulgation process will then follow. If all steps of the AES development process proceed as planned, it is anticipated that the standard will be completed by the summer of 2001.

Last Modified: March 22, 2000

Questions?
Press Contacts

Computer Security Division
National Institute of Standards and Technology

Visitor # 14216 since February 15, 2000.

Ritter's AES Comments

From: ritter@io.com (Terry Ritter)
Newsgroups: sci.crypt
Subject: Re: Letter to NIST: AES Comments
Date: Wed, 30 Apr 1997 21:59:40 GMT
Lines: 63
Message-ID: <3367c046.17037876@news.io.com>
References: <schneier-ya023080002604971232040001@news.visi.com>
<33669857.11517059@nntp-1.io.com> <E9Gny5.78B@cruzio.com>

<this post apparently failed the first time around>

On Wed, 30 Apr 1997 17:15:41 GMT, in <E9Gny5.78B@cruzio.com> in
sci.crypt schlafly@bbs.cruzio.com wrote:

>In article <33669857.11517059@nntp-1.io.com>, ritter@io.com (Terry Ritter) writes:
>> Personally, I doubt that a government standardization process *can*
>> legally require intellectual property "free and clear," or even have a
>> bias against patents to *any* extent whatsoever.
>
>It can and it does. When the govt promotes a standard, it has an
>interest in having one that anyone can use. A patent is just the
>opposite -- it prevents people from using it. Therefore the govt
>has a preference for standards that are available to anyone.

The government has in interest in assuring that a patent used in a
standard will not be used to favor one player against another. Thus,
patents are usually required to be made available on something like a
"non-discriminatory basis."

Once this is assured, patented technology is made "available to
everyone" in pretty much the same way that a Copyrighted book is made
"available to everyone." Which is to say, available to those who can
afford it. We do not see authors being required to contribute *their*
work to the public domain.

In practice, this fear of licensing is nonsense for manufacturers, who
license things all the time, and individuals are rarely an issue. So
the problem is those people who are in business making money (or even
just reputation) with the stuff, but yet for some reason cannot stand
to return any part of their profits to those who have made the profits
possible. *This* is odd.

No patent holder is going to price himself out of the market, nor will
the holder of a standardized patent price it beyond maximum profit,
which is to say, widespread use. It *will* mean that the
manufacturers will not get every dime they can get their hands on.
Big deal.

>> I doubt than *any* author would consider it reasonable to simply
>> *contribute* their copyright to the public domain if their book should
>> be selected for publication. But apparently some authors feel free
>> suggest that patent-holders should contribute *their* work, without
>> any feeling of hypocrisy whatsoever. Odd.
>
>You may think it is odd, but there are a huge number of
>standards, and nearly all of them are in the public domain.

Most standards are defined by *private* standards organizations, which
give various weights to various things. And, of course, most
standards are old.

But even most private standards organizations require only that
patented technology be made available on a non-discriminatory basis.
They generally do *not* require that standards be based on
public-domain technology, or that patents for standards be contributed
to the public domain.

Terry Ritter ritter@io.com http://www.io.com/~ritter/

From: ritter@io.com (Terry Ritter)
Newsgroups: sci.crypt
Subject: Re: Letter to NIST: AES Comments
Date: Wed, 30 Apr 1997 00:56:11 GMT
Lines: 61
Message-ID: <33669857.11517059@nntp-1.io.com>
References: <schneier-ya023080002604971232040001@news.visi.com>
<1997Apr2718.21.49.26877@koobera.math.uic.edu>
<schneier-ya023080002704972002550001@news.visi.com> <336517f7.852271@nntp.netcruiser>
<wtshaw-2904971010130001@207.101.116.61>

On Tue, 29 Apr 1997 10:10:13 -0600, in
<wtshaw-2904971010130001@207.101.116.61> in sci.crypt
wtshaw@itexas.net (W T Shaw) wrote:

>In article <336517f7.852271@nntp.netcruiser>, seward@netcom.ca (John
>Savard) wrote:
>>
>> The AES process is not likely to bring a new, original, and superfast
>> encryption method out of the woodwork; the people who are developing
>> those, in private industry, still hope to make a few bucks from them.
>
>That is where a proprietary implementation phase is so important, to allow
>for some return. I would think such would encourage the best possible
>support in a semi-free market of various platforms and uses.

You were there, and I was not, but I think you should be cautious
about this. My understanding of the proposed "proprietary
implementation phase" is that companies would get the complete cipher
specifications free and clear. This would allow every company to then
compete in the marketplace with their particular trade-secret
implementation. Clearly, this would not reward investment in
cryptographic Research and Development (R&D), and would mean nothing
at all to the cipher designer.

>Back to the submission level, I understand that the government wants
>everything free and clear, but is it reasonable for many to try to deliver
>on that basis? It might not make much of a difference to some, but others
>would like to cover expenses, whether in actual dollars or time.

Personally, I doubt that a government standardization process *can*
legally require intellectual property "free and clear," or even have a
bias against patents to *any* extent whatsoever.

My guess is that similar disputes have played out many times before,
and there are probably laws and rules about intellectual property and
government standards. But then I'm no lawyer.

>Would it be justified to encourage algorithm submissions with prize
>money? That would likely produce many more entries, some good ones that
>would have never been seen, along with some surefire losers as well.

It would have to be one heck of a prize to give up the highest
achievements of a decade of research.

>And, there are patents out there that might be important in making a good
>AES. Certainly, those folks had a financial involvement in getting them.

I doubt than *any* author would consider it reasonable to simply
contribute their copyright to the public domain if their book should
be selected for publication. But apparently some authors feel free
suggest that patent-holders should contribute *their* work, without
any feeling of hypocrisy whatsoever. Odd.

Terry Ritter ritter@io.com http://www.io.com/~ritter/

From: ritter@io.com (Terry Ritter)
Newsgroups: sci.crypt
Subject: Re: Letter to NIST: AES Comments
Date: Fri, 02 May 1997 06:21:57 GMT
Lines: 73
Message-ID: <33697d48.8259943@news.io.com>
References: <schneier-ya023080002604971232040001@news.visi.com>
<schneier-ya023080001059714371400001@news.visi.com> <E9Isqt.u9@cruzio.com>

On Thu, 1 May 1997 20:54:29 GMT, in <E9Isqt.u9@cruzio.com> in
sci.crypt schlafly@bbs.cruzio.com wrote:

>
>>> Terry Ritter ritter@io.com http://www.io.com/~ritter/
>>> But even most private standards organizations require only that
>>> patented technology be made available on a non-discriminatory basis.
>>> They generally do *not* require that standards be based on
>>> public-domain technology, or that patents for standards be contributed
>
>But they prefer public domain technologies.

No, they do not. In fact, they carefully *avoid* having a preference.
Here is part of the ANSI Patent Policy referred to in the AES
announcement:

1.2.11. ANSI Patent Policy - Inclusion of Patents in American
National Standards

There is no objection in principle to drafting a proposed American
National Standard in terms that include the use of a patented item, if
it is considered that technical reasons justify this approach.

If the Institute receives a notice that a proposed American National
Standard may require the use of a patented invention, the procedures
in sections 1.2.11.1 through 1.2.11.4 shall be followed.

1.2.11.1. Statement from Patent Holder

Prior to approval of such a proposed American National Standard, the
Institute shall receive from the identified party or patent holder (in
a form approved by the Institute) either: assurance in the form of a
general disclaimer to the effect that such party does not hold and
does not currently intend holding any invention whose use would be
required for compliance with the proposed American National Standard
or assurance that:

(1) A license will be made available without compensation to
applicants desiring to utilize the license for the purpose of
implementing the standard;
or
(2) A license will be made available to applicants under
reasonable terms and conditions that are demonstrably free of any
unfair discrimination.

[...]

>Neither I nor anyone else in this thread suggested that you be
>required to give away your technology. If you don't want anyone
>using your technology, just don't submit it for the AES.

You should be ashamed to make this argument. I have as much right as
anyone to participate in a government standards process. The freedom
to participate only if I give up my property rights is no freedom at
all. And this would require me to pay to participate in a government
action that non-property holders enter for free.

Similarly, the freedom to participate in a government standards
process which is biased *against* those who do *not* give up their
property rights is also not freedom, not equal treatment under the
law, and would fairly cry out for judicial review.

In any case, it just makes good sense to use the best technology
available. If that turns out to be patented technology, then so be
it. This would not be the end of the world as you know it.

Terry Ritter ritter@io.com http://www.io.com/~ritter/

From: ritter@io.com (Terry Ritter)
Newsgroups: sci.crypt
Subject: Re: Letter to NIST: AES Comments
Date: Fri, 02 May 1997 07:22:32 GMT
Lines: 123
Message-ID: <33698978.11380152@news.io.com>
References: <schneier-ya023080002604971232040001@news.visi.com>
<1997Apr2718.21.49.26877@koobera.math.uic.edu>
<schneier-ya023080002704972002550001@news.visi.com> <336517f7.852271@nntp.netcruiser>
<wtshaw-2904971010130001@207.101.116.61> <33669857.11517059@nntp-1.io.com>
<33674c63.972666@nntp.netcruiser> <schneier-ya023080001059714371400001@news.visi.com>

On Thu, 01 May 1997 14:37:14 -0500, in
<schneier-ya023080001059714371400001@news.visi.com> in sci.crypt
schneier@counterpane.com (Bruce Schneier) wrote:

>In article <33674c63.972666@nntp.netcruiser>, seward@netcom.ca (John
>Savard) wrote:
>
>> ritter@io.com (Terry Ritter) wrote:
>>
>> >I doubt than *any* author would consider it reasonable to simply
>> >*contribute* their copyright to the public domain if their book should
>> >be selected for publication. But apparently some authors feel free
>> >suggest that patent-holders should contribute *their* work, without
>> >any feeling of hypocrisy whatsoever. Odd.
>>
>> You have now made yourself clearer, and Bruce can't complain that what
>> you've just said is an insult to his book. Of course, he might insult
>> your algorithms by replying that they're not the ones he wants to grab
>> for free...
>
>Insulting someone it not the point.

Good. Except that demeaning someone's property is inherently a
personal insult. Stop it now.

>Ritter feels that his algorithms are so
>good that he should be paid for them.

Nope. I feel that if someone wants to benefit from my technology,
they should help cover development costs and a small profit like any
other business transaction.

I have developed and patented and so do own ciphering technology which
can have significant advantages in some applications.

>That's fine. The marketplace can
>speak for itself.

Really? Here is the advice you gave to NIST in message
<schneier-ya023080002302971531110001@news.visi.com>:

#[...]
"Patented algorithms should not be considered, unless the
patent-holder is willing to grant free world wide rights as IBM did
with DES."

The quote is curiously inconsistant with the claim.

>[...]
>I feel the smart move is to give the encryption algorithm away and try
>to leverage the publicity. I did that with Blowfish, which is why Ritter
>is forced to make bizarre comparisons between publishing and cryptography.
>(And I've given away various essays on the net, so the comparison makes
>even less sense. But whatever.)

These so-called "bizarre" comparisons between Copyright and Patent
are a direct result of Schneier -- who makes money by *not* giving his
book content away -- advocating that NIST should *require* or *coerce*
patent holders to give *their* work away. This is not just bizarre,
it is odious.

>[...]
>However, I know several research labs who would simply give
>their algorithms away for AES.

If any company *wants* to give away their work for free, who can
possibly complain? But *that* is not the issue.

Instead, Schneier has advocated that any company which does *not* want
to give their work away should nevertheless be *forced* to do so if
they wish to compete, or *coerced* to do so if they wish to win.

>I don't REQUIRE that everyone give
>their algorithms away.

At first glance we have Schneier's simple definitive statement, but
how can we reconcile that with his actual advice to NIST:

"Patented algorithms should not be considered, unless the
patent-holder is willing to grant free world wide rights as IBM did
with DES."

Just what part of "should not be considered" do we not understand?

>But there will be a plethora of secure and free
>algorithms to choose from, so I expect that those who think they will
>be getting royalties from their algorithm will just go away mad.

If this is directed against me, I can only say that I personally doubt
that *any* individual could possibly win. In this context, perhaps
the best an individual could hope for would be simply to be taken
seriously.

On the other hand, I believe I have a competitive design with valuable
unique features, and if it is *not* taken seriously, I might indeed
get mad and consider my options.

>If there are no free algorithms, we will be forced to use a proprietary one.

Wrong. The first order of business is *not* to decide whether you can
steal someone else's work. I expect that NIST will make this
explicitly clear in their forthcoming rules. Otherwise, their public
meeting has unfortunately already set them up for court challenge.

>Luckily, that won't happen.

Indeed, I am confident that it will *not* happen, because any bias
against patents in government standards is wrong and will not be
allowed to occur.

Terry Ritter ritter@io.com http://www.io.com/~ritter/

From: ritter@io.com (Terry Ritter)
Newsgroups: sci.crypt
Subject: Re: Letter to NIST: AES Comments
Date: Sun, 04 May 1997 21:31:29 GMT
Lines: 99
Message-ID: <336cf698.2295934@news.io.com>
References: <schneier-ya023080002604971232040001@news.visi.com>
<33674c63.972666@nntp.netcruiser> <schneier-ya023080001059714371400001@news.visi.com>
<33698978.11380152@news.io.com> <5kdpmg$lgg@joseph.cs.berkeley.edu>

On 2 May 1997 15:26:24 -0700, in <5kdpmg$lgg@joseph.cs.berkeley.edu>
in sci.crypt daw@joseph.cs.berkeley.edu (David Wagner) wrote:

>
>In article <33698978.11380152@news.io.com>, Terry Ritter <ritter@io.com> wrote:
>>
>> Good. Except that demeaning someone's property is inherently a
>> personal insult. Stop it now.
>>
>
>You know, this is the sort of perspective that can lead to really bad
>science. When scientists get so caught up in their own work that they
>feel personally insulted by scientific criticism of their theories,
>the free flow of ideas is interrupted.

Oh, please. There has been no scientific criticism, and I can hardly
be insulted by what does not exist. Schneier's attitude seems to be
"those ciphers are patented, so nobody should pay any attention to
them," which is not just *UN* scientific, it is actually *ANTI*
scientific. It does seem odd that you would concentrate on the first
which has *not* happened and ignore the greater offense of the latter,
which *has*.

If you really want to know the background, go to the previous postings
in this thread. Read them.

>The fact remains that, all other things being equal, most people will
>probably prefer free unencumbered technology over a patented standard,
>and that's not in any way an attempt to demean or insult you.

All things being equal, most people want a cipher which needs no
initialization, is infinitely fast, uses no resources, is universally
applicable and is guaranteed unbreakable forever.

Oddly, some compromises may have to be made.

The way to do this is *not* to start out saying -- as Schneier has
said -- that patented ciphers should not be considered.

I note that there has been oh-so-much concern about the cost of a
patented cipher standard, but none at all about the cost of a
copyrighted book. If AC were *not* copyrighted, other publishers
could copy the pages and produce an equivalent product for half the
cost or even less. This difference is a Government-protected added
cost to every buyer, yet nobody complains about the use of Copyright
to protect the effort of writing a book. Do people think original
cipher technology grows on trees and so deserves less protection? And
nobody but me complains about the arrogance of someone assuming that
government exists to protect *their* rights, but *not* the rights of
others.

I want to know just how much *less* it will cost the consumer to have
un patented technology. Will the consumer really *see* a price
difference? Or is all this really just about the existing cipher
manufacturers who don't want to have to do anything they didn't have
to do with DES? The Government-supported extra cost that each buyer
of AC must pay would almost certainly pay for multiple patent
licenses.

One of my ciphering approaches is particularly valuable in hardware,
where it is especially fast and efficient. Let's assume that someone
like me *does* have a new idea: If we start out by throwing this away
because it is patented, we can easily end up with something *less*
efficient and *more* costly than the patented approach. In this
sense, a patented cipher actually can be *cheaper* than a free
alternative. But we don't know that until we set aside our
preconceptions.

In such an uncertain coarse trade which presents no guaranteed
marketplace advantage, we need to wonder about someone who claims this
to be an appropriate course of action.

>Don't confuse objective technical commentary on your AES proposal with
>a personal attack on you, ok?

If your "technical commentary" is a hand-wave Opinion that you "know"
my design has problems, you can bet I *will* take it as an attack,
because that is not Science, and cannot be defended against.

If you want to do Science, I don't care about your Opinion; I want to
see facts and detailed reasoning. I want to see claims specified in
enough detail so they can be checked, true or false. *Then* we can
have a discussion based on reality instead of insinuation.

But if you want to make this *fair*, why don't you participate? Why
don't *you* design and enter a cipher? Why don't you make your *own*
decisions about technology and tradeoffs, and fix them in print along
with *your* name?

Then, when you are smacking your lips at the thought of being able to
criticize *my* design, I can be thinking the same about *you*.

Terry Ritter ritter@io.com http://www.io.com/~ritter/

From: ritter@io.com (Terry Ritter)
Newsgroups: sci.crypt
Subject: Re: Letter to NIST: AES Comments
Date: Mon, 05 May 1997 05:40:44 GMT
Lines: 79
Message-ID: <336d72d4.13568636@news.io.com>
References: <schneier-ya023080002604971232040001@news.visi.com>
<schneier-ya023080001059714371400001@news.visi.com> <E9Isqt.u9@cruzio.com>
<33697d48.8259943@news.io.com> <3369e9b4.1050220@nntp.netcruiser>

On Fri, 02 May 1997 13:19:02 GMT, in
<3369e9b4.1050220@nntp.netcruiser> in sci.crypt seward@netcom.ca (John
Savard) wrote:

>ritter@io.com (Terry Ritter) wrote:
>
>>On Thu, 1 May 1997 20:54:29 GMT, in <E9Isqt.u9@cruzio.com> in
>>sci.crypt schlafly@bbs.cruzio.com wrote:
>
>quoting Terry Ritter ritter@io.com http://www.io.com/~ritter/:
>>>>> But even most private standards organizations require only that
>>>>> patented technology be made available on a non-discriminatory basis.
>>>>> They generally do *not* require that standards be based on
>>>>> public-domain technology, or that patents for standards be contributed
>
>>>But they prefer public domain technologies.
>
>>No, they do not. In fact, they carefully *avoid* having a preference.
>>Here is part of the ANSI Patent Policy referred to in the AES
>>announcement:
>
>>1.2.11. ANSI Patent Policy - Inclusion of Patents in American
>>National Standards
>
>>There is no objection in principle to drafting a proposed American
>>National Standard in terms that include the use of a patented item, if
>>it is considered that technical reasons justify this approach.
>
>Er...what if they don't? That sounds like a preference to me:

Really? And if technical reasons don't justify using a public-domain
technology? Is that also a "preference," this time *for* patented
technology?

>there is no objection _in principle_ to including patented items in
>standards, if, for technical reasons, there is _no choice_...

You are inserting your own words into a tightly-constructed policy
statement to impose a meaning that statement does not give on its own.
The policy does not *say* "if there is no choice," nor can we construe
that meaning from anything it *does* say. If you disagree, I want to
see a direct quote from the statement which clearly supports your
interpretation.

To the contrary, the policy directly states the simple fact that a
public standard is being developed does not prevent the inclusion of
patented technology in that standard.

>but in practice, patented items are avoided whenever possible.

Nonsense. Read the quote from my earlier message, the parts you did
not repeat, or read the whole thing on-line as "ANSI Patent Policy."
Nowhere in the document does it say *anything* like:

| "ANSI prefers to use unpatented technology in standards.
| But if no unpatented technology is available, patented
| technology may be used."

Now, the document *could* say that, if it really *meant* what you
claim it means, but the document does *not* say that, and the fact
that it does not, when it could, is telling.

The ANSI Patent Policy is clearly intended to be *unbiased* with
respect to patented technology. It makes clear to patent opponents
that the simple fact that a technology is patented does *not* mean
that it cannot be used in a standard. It makes plain that patented
technology *can* be used, where needed, just like any *other*
technology can be used, where needed. This is *hardly* a preference
for public-domain technology. If it is a preference for anything
here, it is a preference for the best possible technical standard
without regard to patents.

Terry Ritter ritter@io.com http://www.io.com/~ritter/

From: ritter@io.com (Terry Ritter)
Newsgroups: sci.crypt
Subject: Re: Letter to NIST: AES Comments
Date: Mon, 05 May 1997 05:41:53 GMT
Lines: 79
Message-ID: <336d7319.13638437@news.io.com>
References: <schneier-ya023080002604971232040001@news.visi.com>
<1997Apr2718.21.49.26877@koobera.math.uic.edu>
<schneier-ya023080002704972002550001@news.visi.com> <336517f7.852271@nntp.netcruiser>
<wtshaw-2904971010130001@207.101.116.61> <33669857.11517059@nntp-1.io.com>
<33674c63.972666@nntp.netcruiser> <schneier-ya023080001059714371400001@news.visi.com>
<33698978.11380152@news.io.com> <3369ff06.752966@nntp.netcruiser>

On Fri, 02 May 1997 14:57:37 GMT, in <3369ff06.752966@nntp.netcruiser>
in sci.crypt seward@netcom.ca (John Savard) wrote:

>
>ritter@io.com (Terry Ritter) wrote:
>
>>On Thu, 01 May 1997 14:37:14 -0500, in
>><schneier-ya023080001059714371400001@news.visi.com> in sci.crypt
>>schneier@counterpane.com (Bruce Schneier) wrote:
>
>>>If there are no free algorithms, we will be forced to use a proprietary one.
>
>>Wrong. The first order of business is *not* to decide whether you can
>>steal someone else's work. I expect that NIST will make this
>>explicitly clear in their forthcoming rules. Otherwise, their public
>>meeting has unfortunately already set them up for court challenge.
>
>Why, the first order of business *is* to decide whether you already
>have something available to you without spending money -
>
>legitimately, without stealing -
>
>before running out to the store to buy it.

No, the first order of business is to make a comparison of to what
extent each proposed solution will actually solve the problem, and
then what each solution will cost.

But it seems quite *unlikely* that there can *be* any real cost
comparisons in the early stages. Any claimed costs which are bandied
about are likely to be made-up figures designed to win the competition
and which nobody will guarantee in practice. To place such claims on
a par with actual measurements and technical details would be
foolhardy at the very least.

And there seems to be an assumption here that a patented cipher is
necessarily more expensive than a public domain one. This is false.
It is *very* possible for a patented hardware design to be *cheaper*
than a free one, and some people are going to *need* a hardware
solution.

Everybody knows that a software cipher will run faster if we just put
it in hardware. But not everybody understands that some ciphering
architectures improve in hardware *far* *more* than others. Some
designs can have a massively greater bandwidth than other designs,
even if both use similar amounts of similar-technology hardware. The
failure to take advantage of such a design is a lost opportunity
cost, even if the chosen design is free.

>You *do* have a legitimate point, in that NIST should not completely
>prejudge the question of whether a proprietary technology might be
>sufficiently superior to what may be legitimately made available
>without encumbrance, to be worth considering.
>
>To say, however, that NIST should base its decision _only_ on
>technical merit is not correct. A significant benefit, not a slight
>one, is the criterion for accepting a patented technology as the basis
>for a standard.

Sez who? I want to see a quote which backs this up. I researched
this, and posted a quote, and if you have an opposing quote, I want to
see that, and not just your statement to the contrary.

This is a government-sponsored standards process. I claim that
government-granted property rights should not be allowed to be *any*
amount of negative factor in a government standards process,
provided we wish to retain some illusion of equality under law and the
right to own property.

Discrimination against the rights of a small group by the mass is
never pretty, and those who do not speak out may find that their group
is next.

Terry Ritter ritter@io.com http://www.io.com/~ritter/

From: ritter@io.com (Terry Ritter)
Newsgroups: sci.crypt
Subject: Re: Letter to NIST: AES Comments
Date: Tue, 06 May 1997 04:22:57 GMT
Lines: 49
Message-ID: <336eb21e.16211123@news.io.com>
References: <schneier-ya023080002604971232040001@news.visi.com>
<33674c63.972666@nntp.netcruiser> <schneier-ya023080001059714371400001@news.visi.com>
<33698978.11380152@news.io.com> <5kdpmg$lgg@joseph.cs.berkeley.edu>
<336cf698.2295934@news.io.com> <1997May5.145815.7680@mbsks.franken.de>

On Mon, 5 May 1997 14:58:15 GMT, in
<1997May5.145815.7680@mbsks.franken.de> in sci.crypt
m@mbsks.franken.de (Matthias Bruestle) wrote:

>
>Mahlzeit
>
>
>Terry Ritter (ritter@io.com) wrote:
>> I note that there has been oh-so-much concern about the cost of a
>> patented cipher standard, but none at all about the cost of a
>> copyrighted book. If AC were *not* copyrighted, other publishers
>There are some differences here:
>
>- AC is no standard and has no monopoly. AES will be a standard
> and if a firm has a patent on it, it has a monopoly.

A copyrighted book *is* a monopoly on the precise text it contains.
This monopoly exacts a toll in the form of added costs which would not
be present if a competing publisher were allowed to copy the book
exactly and sell it. Such a publisher needs no editors, has almost no
time-to-market delay, and can choose only proven successful products
to virtually eliminate risk. (It has been said that publishing
normally has 9 failures for every success, so the success must earn
back editing and production costs for 10 books.) Such a publisher
could produce far cheaper books. Society prevents this, at a cost
which the customer, of course, must pay.

The reason for the patent monopoly is similar. Without patents we can
have manufacturers who need little or no design group, have low
time-to-market, and can choose only low-risk profitable products.
This sounds like the dream of every manufacturer. But, in the long
run, this is not good for society, even *if* the customer would get a
lower-cost product.

>- Every person can go into a bookstore and buy AC, but it is not
> so easy for a privat person to get a license, or what to do
> if it develops freeware with AES.

So basically you are criticizing the current distribution arrangements
for patents. I agree. But I don't think this is unfixable. It is
not a reason for a blanket prejudice against patenting itself.

Terry Ritter ritter@io.com http://www.io.com/~ritter/

From: ritter@io.com (Terry Ritter)
Newsgroups: sci.crypt
Subject: Re: Letter to NIST: AES Comments
Date: Fri, 09 May 1997 23:39:32 GMT
Lines: 121
Message-ID: <3373b5ac.1003548@news.io.com>
References: <schneier-ya023080002604971232040001@news.visi.com>
<33674c63.972666@nntp.netcruiser> <schneier-ya023080001059714371400001@news.visi.com>
<33698978.11380152@news.io.com> <5kdpmg$lgg@joseph.cs.berkeley.edu>
<336cf698.2295934@news.io.com> <33723bd3.0@ALPHA.RHODES.EDU>

On 8 May 97 20:47:15 GMT, in <33723bd3.0@ALPHA.RHODES.EDU> in
sci.crypt bryan@alejb.nd.rhodes.edu (James Bryan Alexander) wrote:

>[...]
>I have read the previous posts in this thread. And, it may surprise you
>to hear, Terry, that I did not find ANYTHING that was a personal attack
>against you. There were plenty of statements to the effect that using
>a patented algorithm for the standard would be undesirable for several
>reasons, but just because you hold cryptography patents, that doesn't
>make them personal attacks against you. Don't let your insecurities
>and paranoia delude you.

Frankly, I'd really rather get back to work, but this is an issue of
conscience for me.

It would be nice to have most people agree with everything I say (not
that I would know what that was like), I guess. Some of this is just
"Usenet" (people only post *differences*, after all), but I imagine
that many people disagree, and this is not particularly comfortable.
But I call it as I see it.

When all this started -- and I have been here since 1989 when I
announced my Dynamic Substitution stuff after it had been filed with
the PTO -- there were rather few ciphers, and a generally poor
understanding of how one could build an effective cipher. Other
cipher designers began making their work public, so people could use
it and develop from it, which is fine if one is in school or sees this
as a hobby or avocation or a crucial public need. But cipher design
generally takes more than a few weeks, and a general inability to sell
ciphers in a marketplace of free ones means there is no financial
basis to recover a design investment or to support an industry which
would design better ciphers.

At roughly the same time that cipher designers were giving their work
to society, books were being written about cryptography which,
curiously, were *not* given away. Even more curiously, no cipher
designers noticed this difference, or thought it at all strange until,
finally, everyone just assumed that a cipher designer *should* work
for free, while an author should not. Apparently, the "true" cipher
designer works merely for the thrill of seeing his or her name in
print. In print, that is, in books which earn profits for their
authors, but *not* for the many cipher designers who make these books
possible. Note that we don't buy such a book for its immortal prose,
but the author nevertheless insists on a financial basis for his or
her collection effort; writers *don't* give away *their* substantial
works for free.

There is a heavy irony to this, a sort of maturing of understanding
that by giving things away, we fail to develop the financial basis for
an industry which would develop things beyond what we can make now.
This devalues our work, and devalues our respect as cipher designers,
and others take us for fools while using our work in building their
own enterprises for their own profit.

Consider the arrogance of an author who protects his work with
copyright, obviously expecting a return on investment, while insisting
that the cipher designers he supposedly respects should *not* have
similar protection if they want to even *compete* for inclusion in a
public standard. This *is* an insult, a direct slap in the face of
every designer who considers cipher design a serious activity which
deserves a return on investment in our capitalist economy.

The idea that patented things are *necessarily* more costly than free
things is simply false. Sometimes, as in hardware, patented things
can be *cheaper* than free designs, if they use fewer resources. More
often, the price of the cipher is hidden in the product, and a
patented cipher will *not* increase the price. So the idea that
patents are necessarily costly is an unwarranted bias, a *prejudice*.
And, like most prejudices, it is not good policy, not (just) on moral
grounds, but also on the *practical* grounds that making false
assumptions does not lead to the best decisions.

This is a free society, and there are cipher designers who give their
work away free, and those who don't. However, this is also a
capitalist society, based on profit in the marketplace. Demeaning
those who would make such a profit is an insult to business itself.
For an author to suggest that the class of designers who have the
nerve to consider their work as important as that of an author should
be discriminated against is *inherently* an insult.

I expect my government to equally respect the rights of all of us, and
patents clearly *are* property rights. The idea that someone should
try to induce my government to discriminate against me on the basis of
my *rights* in a publicly financed process is morally *wrong*, an
insult to me, and a mischief to a government process.

I think that cipher designers who try to profit from their work
deserve a public apology for these insults. We deserve actions which
would correct this injustice, and a promise to support the concept of
a continuing business of cipher design functioning in ways consistent
with our economic system.

>[...]
>Well, maybe you can afford to pay patent royalties, but if I am implementing
>a crypto system for localized use, like within a small company, I cannot
>afford to pay patent royalties, and don't really have the time or inclination
>to do the paperwork to get a patent license.

If you can afford to buy the equipment to run the cipher, you can
afford to pay for the cipher. Period.

It is more realistic for most people, however, to simply buy a program
which contains the cipher, and also naturally includes a license for
use hidden in the purchase cost.

>I'd just as soon use something
>in the public domain and not worry about patent hassles.

My particular cipher technology has certain advantages in certain
applications, and in some cases, these advantages can be crucial. But
if one is just using my technology as one would use other ciphers, I
doubt that we would see much of an advantage. So if you don't need
what I have, or find it particularly advantageous, don't use it.

Terry Ritter ritter@io.com http://www.io.com/~ritter/

From: ritter@io.com (Terry Ritter)
Newsgroups: sci.crypt
Subject: Re: Letter to NIST: AES Comments
Date: Sun, 11 May 1997 20:17:53 GMT
Lines: 47
Message-ID: <33762964.11711744@news.io.com>
References: <3373b5ac.1003548@news.io.com> <E9yGBq.199@cruzio.com>
<01bc5d77$bbab7a40$27c10c26@i-charlej9.interramp.com> <5l4cp6$pbt@news.ox.ac.uk>

On 11 May 1997 12:07:02 GMT, in <5l4cp6$pbt@news.ox.ac.uk> in
sci.crypt patrick@gryphon.psych.ox.ac.uk (Patrick Juola) wrote:

>[...]
>Let me ask an analogous question. Suppose both algorithms were unpatented,
>but algorithm A were 33% faster. Should we choose algorithm A?
>
>How about if algorithm A were 33% faster, but required four times as
>much memory? Should we still choose algorithm A?

The weighting of various features will no doubt be a subsequent issue.

>Availability (and cost) of the algorithm is, and should be, a criterion
>like any other one.

Well, *availability* -- nondiscriminatory licensing or some such -- is
the one thing which *is* required of a patented standards candidate.

But *cost* simply *cannot* be determined. Final to-the-user cost of a
cipher will probably be the function of several intermediates, with
the cipher only a small part of the ultimate product. Costs like
these simply cannot be accurately described in terms of their affect
on the user. Does each API which Microsoft adds to their operating
system really *have* a known effect on the OS price?

>>To put it bluntly, Terry asks (IMHO, of course) if the NIST is after the
>>best algorithm possible (whatever "best" may mean here) what difference
>>does it make if the algorithm happens to be patented?
>
>The same difference that speed, memory overhead, &c.... makes.

Of these is that practical difference, then simply having general
principles, and the people who are in charge of the world is
reasonable basis for a decision.

Terry Ritter - ritter@ix.net - http://www.io.com/~ritter/

From: ritter@io.net (Terry Ritter)
Newsgroups: sci.crypt
Date: 1996/12/31 00:00:00
Subject: Re: Cryptanalysis
Author: ritter@io.net (Terry Ritter)
Organization: Cryptography Consulting and Development
Reply-To: ritter@io.net (Terry Ritter)
Contact: ritter@io.net (Terry Ritter) - http://www.io.com/~ritter/
Copyright: Copyright (c) 1996 Terry Ritter

On 30 Dec 1997 12:00:00 -0700, in <3F1ZqMnh@io.net> in sci.crypt ritter@io.net (Terry Ritter) wrote:

In article <4ddq8q$ 1ed@io.net>, in sci.crypt Terry Ritter ritter@io.net wrote:
> Cryptanalysis is a very specialized field, and it is difficult to find people who have both general cryptanalytic skills and specialized knowledge of the particular cipher being attacked.

There are very rare to period a time but and since use of research. Cryptanalysis is a very specialized field, and it is difficult to find people who have both general cryptanalytic skills and specialized knowledge of the particular cipher being attacked.

Block and stream, and of and matter.

Yes, from publication source of the time in the end.

In any case, the art is unfortunately, but it is not specific to Block and stream, and of and matter. Cryptanalysis is a very specialized field, and it is difficult to find people who have both general cryptanalytic skills and specialized knowledge of the particular cipher being attacked.

Block and stream, and of and matter.

Yes of the time a problem with this practice, and single and Block and stream, and of and matter. Cryptanalysis is a very specialized field, and it is difficult to find people who have both general cryptanalytic skills and specialized knowledge of the particular cipher being attacked.

My problem is that the practice of cryptanalysis is very specialized. Nevertheless, the practice is useful to the world without the need for people who have both skills. Cryptanalysis is a very specialized field, and it is difficult to find people who have both general cryptanalytic skills and specialized knowledge of the particular cipher being attacked.

The cryptanalysis I draw from is from the public domain cipher. Cryptanalysis is a very specialized field, and it is difficult to find people who have both general cryptanalytic skills and specialized knowledge of the particular cipher being attacked. Cryptanalysis is a very specialized field, and it is difficult to find people who have both general cryptanalytic skills and specialized knowledge of the particular cipher being attacked.

Yes, the public domain cipher. I don't know the publication.

Yes, the public domain cipher. I don't know the publication. Cryptanalysis is a very specialized field, and it is difficult to find people who have both general cryptanalytic skills and specialized knowledge of the particular cipher being attacked.

Block and stream, and of and matter

S-Box Design: A Literature Survey

Research Comments from Ciphers By Ritter

Terry Ritter

Many block ciphers based on the DES have relied on the apparent application of confusion and diffusion. Typically, confusion is provided by some form of substitution ("S-boxes"), by the nonlinear combination of key and data, and diffusion is provided by some form of transposition (or "P-boxes") which rearrangs the bits so that they can be processed by further substitution at all.

There has not been as much research which would distinguish between "bad" and "good" substitutions, and few techniques to construct "good" substitutions. But since weakness measures are related to cryptanalytic attacks, we can at least avoid the worst substitutions. Unfortunately, we do not know how to make the best substitutions, or even how to know that we have made them.

Accordingly, this reviewer has a bias that considers substitutions which have been created by expected or have optimum strength against whatever is being measured, they can be expected to have some weakness against other attacks. Rather there is no systematic design, there can be no systematic weakness. And when S-boxes are chosen at random, everyone can be fair and say that S-boxes "big-bang" at present. Keying the S-boxes invariably takes time, but some authors count this as an advantage in strong attacks.

This page has links to some of the literature that describes the concept of "nonlinear".

Contents

- Intro
  - Friend gives us the concept of "avalanche." Its surprisingly timeless comments, but this is the context of trying to prove individual ciphers.
  - Kam and Davida give us the concept of "complementary." They are concerned with the particular structure which we now call a Substitution - Permutation (S-P) cipher. Most substitutions in normally-keyed ciphers are fixed, but they have a chance to adapt to other ciphering elements.
  - Gorden and Retkin cause the number of randomly chosen S-boxes which contain linear relationships. These results were updated by Youssef and Tavares.
  - Sorkin explains a S-P cipher where the key-dependent S-boxes are chosen from a set of fixed S-boxes, so it is clear that there is no "back door."
- Intro
  - Webster and Tavares review "complementary" and "avalanche" and give us the Strict Avalanche Criterion (SAC).
  - Pieprzyk and Finkelstein discuss the expected nonlinearity of S-boxes chosen at random.
  - Adams relates strict avalanche to the Bit Independence Criterion (BIC).
  - Meier and Staffelbach give us "perfect nonlinearity" and relate this to definitions in terms of the strict avalanche criterion (SAC).
  - Pieprzyk gives us the "strict avalanche" and "perfect nonlinearity" criteria in terms of Boolean functions.
  - Pieprzyk and Finkelstein deal with the design and construction of non-linear permutations (S-boxes).
  - Pieprzyk discusses the nonlinearity of Boolean functions.
  - Lloyd investigates connections between the SAC, balance, and correlation immunity.
  - Preneel, Van Leekwijck, Van Linden, Govaerts and Vandewalle generalize the SAC and perfect nonlinearity to a Propagation Criterion of degree k. The Walsh-Hadamard transform is used.
  - Nyberg gives us "perfect nonlinearity" and a construction for such S-boxes.
  - Lloyd gives us a class of S-boxes that are easy to show meet the Propagation Criterion of degree one.
- Intro
  - Forsyth, Jovanovic and Pieprzyk discuss the information leakage in S-boxes, and the Strict Avalanche Criterion (SAC).

The Story of Combiner Correlation: A Literature Survey

Research Comments from Ciphers By Ritter

Terry Ritter

Once upon a time, a stream cipher was little more than a linear feedback shift register (LFSR) and a simple exclusive-OR combiner. The fly in the ointment was the known-plaintext attack, which quickly bypassed the simple combiner to reveal the sequence from the LFSR. (The Berlekamp-Massey algorithm will recover the unknown state of a simple n-bit LFSR, *and* its feedback polynomial, with just 2n known bits.) Because of the weakness of the data-confusion combiner and the simplicity of the LFSR, even a degree-32 LFSR, with a cycle length of $2^{32} - 1$ or 4×10^9 steps, can be penetrated by knowing (or guessing) just 8 plaintext characters (64 bits).

The need to generate a more "complex" sequence led to the idea of using multiple LFSR's and somehow mixing them so that the ultimate complexity was the product of the individual complexities. This led to the construction of new confusion combiners, and to the analysis of those combiners. Eventually, Siegenthaler found that input-output correlations could be used to attack many of these combiners, even *without* using known-plaintext. This led to a sequence of developments producing stronger combiners, and more effective attacks. This process is the story of combiner correlation.

Contents

- 1965
  - "Before the Beginning", MacLaren and Marsaglia combine random number sequences with a table.
- 1972
  - The State-of-the-Open-Art seems to be a simple LFSR with a simple exclusive-OR combiner, but that is attackable with only 2n known bits.
- 1973
  - *Combiner:* Geffe proposes combining the sequences from two LFSR's by using a third LFSR to select or *multiplex* between the two.
- 1977
  - *Combiner:* Pless proposes combining the sequences from two LFSR's by feeding each into the J or K input of a J-K Flip Flop, and also deleting alternate output bits from the FF.
- 1979
  - *Attack:* Rubin shows that at least one Pless "Arrangement" can be broken at practical effort.
- 1984
  - *Combiner:* Bruer proposes that the single output bits from each of three or more LFSR's be combined by integer addition, and the output be taken from a threshold function.
  - *Attack:* Siegenthaler proposes a form of "divide and conquer" attack allowing each LFSR to be worked on separately. Siegenthaler finds this to be possible when the combiner has some amount of correlation between an input sequence and the output.
  - *Attack:* Retter shows that a known-plaintext attack can break a MacLaren-Marsaglia combiner.
- 1985
  - *Attack:* Siegenthaler continues on the "correlation attack" and presents graphs showing the performance of a *ciphertext only* version. Siegenthaler claims to be able to break Geffe, Pless, *and* Bruer, and presents graphs showing the performance of the attack, but no explicit algorithms.
  - *Combiner:* Rueppel proposes to combine the single bit output of each of two LFSR's, plus a saved "carry" bit, by integer addition. The combiner output is the "sum" output from the adder, and the carry bit is set from the "carry" output. This is the "summation cipher."
  - *Theory:* Siegenthaler presents the general form of correlation-immune combiners as a combinatoric circuit with delayed feedback.
  - *Attack:* Retter shows that MacLaren-Marsaglia combiners support an effective divide-and-conquer attack, and so are not cryptographically effective.
  - *Attack:* Siegenthaler addresses the situation of a single LFSR which is tapped at several places to provide multiple inputs for a nonlinear combining function.
- 1986
  - *Theory:* Maurer (in **German**) shows that the "summation cipher" is correlation-immune.
  - *Theory:* Rueppel writes the book on stream ciphers.
  - *Production:* Ciarcia presents a fully-realized design, and kits for that design, for a supposedly-secure simple stream cipher. (Later, Pearson shows how to efficiently break it.)
- 1987
  - *Attack:* Mund, Gollmann and Beth improve the efficiency of a correlation attack by using the Walsh-Hadamard transform.
- 1988
  - *Theory:* Guo-Zhen and Massey show that, for any *memoryless* combining function, combiner correlation can be computed by a Walsh transform of the combining function.
  - *Attack:* Meier and Staffelbach describe algorithms and give results for their own correlation attack.
  - *Attack:* Zeng and Huang approach the correlation attack as an *error-correction* problem, where the (hidden) ideal LFSR sequence has been partially corrupted by another sequence.
  - *Attack:* Pearson shows how to break the Ciarcia design.
- 1989
  - *Attack:* Forre modifies an algorithm from Meier and Staffelbach to attack a different form of combined generator. In this form, a single LFSR is tapped at several places to provide multiple inputs for a nonlinear combining function.
- 1990
  - *Attack:* Meier and Staffelbach attack and break a "summation cipher" which uses two sizable LFSR's. They show that the same approach would not work with more than two LFSR's.
  - *Attack:* Mihaljevic and Golic give a detailed algorithm (extended from Meier and Staffelbach "Algorithm B") and its performance.
  - *Attack:* Zeng, Yang and Rao find a detailed new approach which they call a "linear syndrome" algorithm. This is an error-correcting approach which also is applied iteratively.
  - *Attack:* Staffelbach and Meier continue to review the "summation cipher," and find that the carry is asymptotically *balanced* for even numbers of LFSR's, and *biased* for odd numbers of LFSR's. In general, the more LFSR's the better, and, thus, their proposed three-LFSR summation cipher is weak. cipher
  - *Three-Input Combiner:* Golic and Mihaljevic investigate the linear complexity of a LFSR sequence which is locally re-ordered by memory and separate read and write LFSR's.
  - *Combiner:* Ritter presents the concept of a *dynamic* function as a reversible combiner, so being useful both for confusion-confusion *and* data-confusion combining.
- 1991
  - *Attack:* Chepyzhov and Smeets present a new algorithm for LFSR state recovery from noisy data.
  - *Theory:* Camion *et. al.* establish a link between *correlation-immune* functions, and *orthogonal arrays* by way of Algebraic Coding Theory.
  - *Attack:* Mihaljevic and Golic continue to improve and test their iterative algorithms.
  - *Survey:* Zeng *et. al.* survey the situation with respect to stream ciphers, LFSR's and combiners.
- 1992
  - *Attack:* Mihaljevic and Golic present a different iterative algorithm in some detail.
- 1994
  - *Attack:* MacKay presents yet another attack algorithm, including C-style pseudo-code, using "variational free energy minimization."
  - *Attack:* Mihaljevic presents yet another attack on LFSR mixing.
  - *Attack:* Menicocci shows that the sequence generated by the Golic and Mihaljevic Variable-Memory BSG is cryptographically weak.
  - *Attack:* Golic presents a general theory of modelling various LFSR-based generators.
- 1995
  - *Attack:* Golic *et. al.* presents several forms of error-correcting algorithm in detail, and compares results.
  - *Attack:* Klapper and Goresky show that the summation cipher is in fact the addition of "2-adic" numbers, and present a sequence reconstruction algorithm analogous to Berlekamp-Massey.

1965 -- Before the Beginning

MacLaren, M. and G. Marsaglia. 1965. Uniform Random Number Generators. *Journal of the Association for Computing Machinery.* 12(1): 83-89.

"In attempting to improve on the congruential generators, we tried combining two of them. This gave a generator which seems to be better than either of the two congruential generators, but it has the disadvantage of being slower."

"Suppose the first number U_1 for a congruential generator . . . is picked at random. Then the sequence U_1, U_2, ... may be considered a sequence of random variables. Moreover, each U_i will be uniform on the set of numbers in [0,1] which can be represented exactly in the computer. However, the different U_i are not independent, and it turns out that the distribution of an *n*-tuple (U_1, ..., U_n) may be quite far from the correct distribution. To improve the distribution of *n*-tuples, we propose using two different generators . . . and having one shuffle the sequence produced by the other."

Although not originally developed for cryptography, the MacLaren-Marsaglia combiner was broken by known plaintext attack, and by divide and conquer.

1972 -- State-of-the-Open-Art

One example is an article by an Applications Engineer at National Semiconductor:

Twigg, T. 1972. Need to keep digital data secure? *Electronic Design.* 23: 68-71. November 9. (Also see: Smith, M. 1973. Correction suggested in encoding article. *Electronic Design.* 9: 7. April 26.)

Twigg, an Applications Engineer at National Semiconductor, proposes to encrypt data with a pseudorandom sequence generated by a shift-register (SR) and exclusive-OR gates. (This is known as a "linear feedback shift register" or LFSR.) The SR is composed of 7474 TTL "D" flip-flops. The feedback polynomial is arbitrarily selectable using a switch at each stage.

But following this article -- in *the very same issue* -- is the moderately-famous article by Meyer and Tuchman at IBM:

Meyer, C. and W. Tuchman. 1972. Pseudorandom codes can be cracked. *Electronic Design.* 23: 74-76. November 9.

"In general, the code of any n-stage code generator, with arbitrary feedback . . . can be broken with any 2n bits of clear and corresponding enciphered text. Breaking the code consists of determining the switch settings and the initial states of the flip-flop stages. Once these conditions are known, the complete text can be deciphered."

They then gave a matrix-based reconstruction.

1973 -- Geffe (U.S.A.)

Probably after digesting the irony presented by *Twigg* and *Meyer-Touchman* in the same issue, Geffe published his designs for stronger stream ciphers:

Geffe, P. 1973. How to protect data with ciphers that are really hard to break. *Electronics.* January 4. 99-101.

Geffe presented a general design using a 33-bit linear feedback shift register (LFSR) with key-selectable feedback taps. But his main innovation was an attempt to create a nonlinear keystream by combining three separate LFSR's. One LFSR was used solely to select between two other LFSR's, as follows:

```
A ------|\
        |&)--+
     +--|/   +--)\                    -         IF C THEN
C ---+          )+)--- Z    Z = AC + BC   or       Z := A
     +-o|\   +--)/                              ELSE
        |&)--+                                     Z := B;
B ------|/
```

Note that the Geffe combiner is actually a *multiplexer,* which is now a common structure used to select between two inputs. Let's look at some simple probability results from the Geffe combiner:

A	B	C	Z	A=Z	B=Z	C=Z
0	0	0	0	1	1	1
0	0	1	0	1	1	0
0	1	0	1	0	1	0
0	1	1	0	1	0	0
1	0	0	0	0	1	1
1	0	1	1	1	0	1
1	1	0	1	1	1	0
1	1	1	1	1	1	1
			---	---	---	---
			50%	75%	75%	50%

First we note that the Geffe combiner does indeed produce a balanced result. That is, assuming the input sequences are uncorrelated and each has an equal number of 1's and 0's, the output will also have and equal number of 1's and 0's.

But then we see that, surprisingly, whatever the output value Z, the probability is 75% that input A has that same value, and the same can be said for B. This is an input-to-output correlation or "information leak," and was eventually used to break through the combiner to find the state of the individual SR's.

The Geffe combiner is broken by Siegenthaler as a ciphertext only attack.

1977 -- Pless (U.S.A.)

Pless, V. 1977. Encryption Schemes for Computer Confidentiality. *IEEE Transactions on Computers.* C-26(11): 1133-1136.

Pless proposes that we feed the bit output from each of two LFSR's into the J and K inputs of a J-K Flip Flop, and then also delete alternate output bits with an "alternator." The J-K Flip Flop is the actual Pless combiner:

```
     +----+                 -    -
A ---|J  Q|--- Q[t+1] = Q[t]K + Q[t]J
     |    |
     |    |
B ---|K   |
     +----+
```

And here are the probability results:

Q[t]	A	B	Q[t+1]	A=Q[t+1]	B=Q[t+1]
0	0	0	0	1	1
0	0	1	0	1	0
0	1	0	1	1	0
0	1	1	1	1	1
1	0	0	0	1	1
1	0	1	1	0	1
1	1	0	0	1	1
1	1	1	1	0	1
			---	---	---
			50%	75%	75%

We note that the Pless combiner does produce a balanced result. But whatever the output value Q[t+1], the probability is 75% that input A has that same value, and the same can be said for B. This is an input-to-output correlation that was used to break through the combiner and find the state of the individual SR's.

After arguing for "alternators" in "Arrangements" B and C, Pless fails to include them in Arrangement D, and this is the configuration which was broken by Rubin in a known-plaintext attack. The Pless combiner was also broken by Siegenthaler in a ciphertext-only attack.

1979 -- Rubin (U.S.A.)

Rubin, F. 1979. Decrypting a Stream Cipher Based on J-K Flip-Flops. *IEEE Transactions on Computers.* C-28(7): 483-487.

(Also reprinted in *Cryptologia,*, 5(1), Jan. 1981 and *Cryptology: Yesterday, Today and Tomorrow,* 1987, Deavours, C *et. al.*, eds., 283-293.)

"*ABSTRACT:* Pless has proposed a stream cipher based on J-K flip-flops that uses eight linear shift registers with feedback, having a combined length of 97 bits, four J-K flip-flops, and a four-stage cycling counter. The cipher has 2.54×10^{51} initial states (keys), and generates a presumably pseudorandom stream whose period is 1.52×10^{29} bits. Despite these impressive statistics, it is computationally feasible to solve such a cipher with a known-plaintext attack, using as few as 15 characters."

1984 -- Bruer (Sweden)

Bruer, J. 1984. On Pseudo Random Sequences as Crypto Generators. *Proceedings of the International Zurich Seminar on Digital Communications* 157-161.

Bruer proposes that the single output bits from each of three or more LFSR's be combined by integer addition, and the output be taken from a threshold function. This combining is also broken by Siegenthaler in a ciphertext only attack.

1984 -- Siegenthaler (Switzerland)

Siegenthaler, T. 1984. Correlation-Immunity of Nonlinear Combining Functions for Cryptographic Applications. *IEEE Transactions on Information Theory.* IT-30: 776-780.

"Pseudonoise generators for cryptographic applications consisting of several linear feedback shift registers with a nonlinear combining function have been proposed as running key generators in stream ciphers." The purpose of the nonlinear combining function *f* . . . is to make the keystream difficult for the cryptanalyst to predict." "However, if the function *f* is not properly chosen, a cryptanalyst may make a selective attack on each subkey . . . ; this can be performed by correlating the ciphertext with the sequence generated by subgenerator S_i for each choice of K_i.

"In general, to make the generator . . . resistant to a correlation attack, one should ensure that there is no statistical dependence between any small subset of the *n* subgenerator sequences and the keystream sequence."

"Let$\boldsymbol{X}_j = (X_{1j}, X_{2j}, \ldots, X_{nj})$ be the *n*-tuple of subgenerator output digits at time *j*. We shall say that the combining function *f* is *m*th-order correlation-immune if every *m*-tuple obtained by choosing *m* components from $\boldsymbol{X}_j$ is statistically independent of Z_j for all *j* = 1, 2, 3, ..."

1984 -- Retter (U.S.A.)

Retter, C. 1984. Cryptanalysis of a MacLaren-Marsaglia System. *Cryptologia.* 8(2): 97-108.

"The research and development groups at Data General do much of their work on a large network of interconnected minicomputers." "For this reason, various file encryption programs have been developed. The early versions were trivial, but by 1980 a program was in use which its author claimed to be 'virtually unbreakable short of exhaustive search.' Since the key was 31 bits, exhaustive search might have been possible, but on the available minicomputers it would have taken days of CPU time even with known plaintext. The system proved to be far less secure, and can usually be broken in minutes using just a guess about the plaintext."

"This algorithm is a version of the MacLaren-Marsaglia algorithm, which Knuth [2] contends 'will satisfy virtually anyone's requirements for randomness." "The method of attack used was the known-plaintext attack."

1985 -- Siegenthaler (Switzerland)

In a manuscript generally contemporaneous with his previous article, Siegenthaler presents graphs showing the *performance* of his correlation attack. In particular, he takes on the Geffe combiner and shows that it supports a correlation *ciphertext-only* attack.

Siegenthaler, T. 1985. Decrypting a Class of Stream Ciphers Using Ciphertext Only. *IEEE Transactions on Computers.* C-34: 81-85.

1985 -- Rueppel (Switzerland)

Rueppel, R. 1985. Correlation Immunity and the Summation Generator. *Advances in Cryptology -- CRYPTO '85.* 260-272. Springer-Verlag.

". . . integer addition, when viewed over GF(2), defines an inherently nonlinear function with memory whose correlation-immunity is maximum."

". . . consider a classical running-key generator for use in a stream cipher system. Such a running-key generator consists of N driving linear feedback shift registers (LFSRs) and some nonlinear function operating on the N output sequences in order to produce the running-key." "Unfortunately, for . . . memoryless combining functions *f* there exists a *tradeoff* between the attainable nonlinear order and the attainable level of correlation-immunity."

". . . one bit of memory suffices to obtain nonlinear combiners that are maximally correlation-immune and have maximum nonlinear order at the same time."

Rueppel then goes on to propose that the one bit output from each of two LFSRs be combined by integer addition along with a "carry" memory bit. The carry is a one-bit delay for the carry output of the adder, and the sum output is the combined output.

1985 -- Siegenthaler (Switzerland)

Siegenthaler, T. 1985. Design of Combiners to Prevent Divide and Conquer Attacks. *Advances in Cryptology -- CRYPTO '85.* 273-279. Springer-Verlag.

"A common form of running key generators for use in stream ciphers consists of n driving sources and some combiner. We assume . . . that a finite state machine (FSM) combines the n input sequences . . . into an output sequence . . . ."

"A cryptanalyst possibly tries to break the above system by breaking the individual subkeys of the n sources. To prevent such divide and conquer attacks, the symbols generated by the FSM should be statistically independent on the symbols of one (or several) input sequences."

1985 -- Retter (U.S.A.)

Retter, C. 1985. A Key-Search Attack on MacLaren-Marsaglia Systems. *Cryptologia.* 9(2): 114-130.

"The idea of combining multiple pseudo-random number generators in order to produce a more secure keystream sequence has been proposed in various forms [5, section 6.4]. Most of these methods are intended to create nonlinear sequences by using linear generators, since linear sequences are easily invertible."

"The MacLaren-Marsaglia algorithm [1,2,3] is a somewhat more complex method of combining two generators. It stores a collection of previous values from one generator in a table, and uses the other generator to select which value to output from the table at each cycle."

"It is the purpose of this paper to show that the algorithm can be attacked by searching for the key to one of its generators, while ignoring the other." "Therefore, the amount of computation required to break the combined keystream generator is only about twice what would be required if one of the input generators had been used to generate the keystream directly."

1985 -- Siegenthaler (Switzerland)

Siegenthaler addresses the situation of a single LFSR which is tapped at several places to provide multiple inputs for a nonlinear combining function.

Siegenthaler, T. 1985. Cryptanalysts Representation of Nonlinearly Filtered M-Sequences. *Advances in Cryptology: EUROCRYPT '85.* 103-110. Springer-Verlag.

"Abstract: A running key generator consisting of a maximum-length (ML) linear feedback shift register (LFSR) and some nonlinear feedforward state filter function is investigated. It is shown how a cryptanalyst can find an equivalent system in a ciphertext-only attack. The analysis uses a Walsh orthogonal expansion of the state filter function and its relation to the crosscorrelation function (CCF) between the ML-sequence and the produced running key."

1986 -- Maurer (West Germany?)

Maurer, U. 1986. On the Linear Complexity and Correlation Immunity of the Summation Cipher. *Mitteilungen AGEN* 44: 5-12. (Dec. 1986) (**In German.**)

"**Abstract.** The Summation cipher, introduced by Massey and Rueppel, uses integer addition with carry as the combining function in the key stream generator for an additive stream cipher." "The correlation immunity of the summation cipher is proved. If the driving shift-registers are short, the resulting leakage of the input sequences through the combiner is shown to be the effect of the periodic repetition of a biased input sequence.

1986 -- Rueppel (Switzerland)

Rueppel releases not just a single article, but an entire book on stream cipher design.

Rueppel, R. 1986. *Analysis and Design of Stream Ciphers.* Springer-Verlag.

1986 -- Ciarcia (U.S.A.)

Ciarcia, S. 1986. Build a Hardware Data Encryptor. *Byte.* September. 97-111.

"This easy-to-build device is extremely difficult to crack."

"The technique used here is to combine the output of two pseudorandom sequencers to produce one pseudorandom stream." "The length of the bit stream generated by the top four shift register [devices] in figure 3 is $2^{31} - 1$ or 2,147,483,647. The length of the bit stream of the lower three shift register [devices] is $2^{23} - 1$ or 8,388,607."

This design is broken by Pearson.

1987 -- Mund, Gollmann and Beth (West Germany)

Mund, S., D. Gollmann and T. Beth. 1987. Some remarks on the cross correlation analysis of pseudo random generators. *Advances in Cryptology -- EUROCRYPT '87.* 25-35. Springer-Verlag.

"Siegenthaler has shown how cross-correlation techniques can be applied to identify pseudo random generators consisting of linear feedback shift registers and a scrambling function." "It is possible to speed up this attack by using the Walsh-Hadamard Transform to compute simultaneously the cross correlation between (c^n) and the outputs of all possible initial states of the given register."

"We show that there exists a trade-off between the dimension of the Hadamard matrix and the number of bits required to compute the cross correlation analysis."

1988 -- Guo-Zhen and Massey (China and Switzerland)

Massey himself weighed in with a way to test (some) combining functions for correlation:

Guo-Zhen, X. and J. Massey. 1988. A Spectral Characterization of Correlation-Immune Combining Functions. *IEEE Transactions on Information Theory.* 34(3): 569-571.

"*Abstract* -- It is shown that a Boolean combining function *f(x)* of *n* variables is *m*th-order correlation immune if and only if its Walsh transform *F(w)* vanishes for all *w* with Hamming weights between 1 and *m,* inclusive. ..."

"A binary [i.e., GF(2)-valued] random variable is said to be *balanced* if it is equally likely to take on the values 0 and 1. Siegenthaler [2] has defined the combining function *f(x)* to be *mth-order correlation immune* if the random variable $Z = f(X_1, X_2, ..., X_m)$ is statistically independent of every set of *m* random variables chosen from the balanced and independent binary random variables X_1, X_2, ..., X_N."

"*Theorem:* The Boolean combining function *f(x)* for *n* binary variables is *m*th-order correlation immune, where 1 <= *m* <= *n*, if and only if its Walsh transform satisfies *F*(w) = 0, for 1 <= *W(w)* <= *m*."

(Here *W(w)* is the Hamming *weight;* that is, the number of 1's in the vector.) The result basically says that if we perform a Walsh transformation on the sequence produced by stepping through the combining function, we can tell how correlation immune that function is. (Unfortunately, this cannot hold for dynamic functions.)

1988 -- Meier and Staffelbach (Switzerland)

Meier and Staffelbach give us two actual algorithms ("A" and "B") for developing LFSR state behind the combiner.

Meier, W. and O. Staffelbach. 1988. Fast Correlation Attacks on Stream Ciphers. *Advances in Cryptology -- EUROCRYPT '88.* 301-314. Springer-Verlag.

"This leads to new design criteria for stream ciphers:"

1. "Any correlation to a LFSR with lest than 10 taps should be avoided."
2. "There should be no correlation to a general LFSR of length shorter than 100 (especially when the feedback connection is assumed to be known)."

"It is remarkable that the importance of the number of LFSR taps for the correlation analysis was not recognized in the cryptographic literature so far."

1988 -- Zeng and Huang (China)

Zeng, K. and M. Huang. 1988. On the Linear Syndrome Method in Cryptanalysis. *Advance in Cryptology -- CRYPTO '88.* 469-478.

"Suppose the cryptanalyst has at his hand a sufficiently long segment of the binary sequence

B = A + X,

"where A is a linear sequence with known feedback polynomial f(x) and x is a sequence with unknown or very complicated algebraic structure, but is sparse in the sense that, if we denote its signals by x(i), i > 0, then we shall have

s = prob(x(i) = 1) = 1/2 - e, 0 < e < 1/2 .

"We call s the error rate of the sequence A in the sequence B."

"One way for tackling this problem is to make use of the ideas of error correction . . . ."

1988 -- Pearson (U.S.A.)

Pearson, P. 1988. Cryptanalysis of the Ciarcia Circuit Cellar Data Encryptor. *Cryptologia.* 12(1): 1-10.

"This paper abstracts the encryption algorithm presented in [the Ciarcia] article, and recasts it in matrix notation for further analysis."

"The two LFSRs in this design are binary registers of length 31 bits and 23 bits, resulting in a total of 2 possible internal states for the keystream generator. This large number (2 = 1.8×10^{16}, more impressively pronounced as "eighteen million billion") effectively eliminates the possibility of finding the key by exhaustive search."

"Here is how equation (2) would be used in a known-plaintext attack. First, 54 consecutive bits of the keystream must be calculated by XORing 54 bits of intercepted ciphertext with the corresponding 54 bits of plaintext. (For example, if the attacker knows that the transmission starts with 'Dear Sir' in 8-bit ASCII, he already has 64 plaintext bits -- 8 characters of 8 bits each.) Next, these 54 bits are arranged into a column vector $\mathbf{K}_1$ and multiplied by $\mathbf{D}_{-1}$ to yield $\mathbf{A}_0$."

"Finally, this $\mathbf{A}_0$ is loaded into a keystream generator, where it will generate first the 54 keystream bits from which it was calculated, then the keystream bits needed to decrypt the rest of the intercepted message."

"If the attacking cryptanalyst doesn't know the first 54 bits of plaintext, there are other avenues still open."

1989 -- Forre (Switzerland)

Where previously we were concerned with a nonlinear combining of multiple separate LFSR's, here Forre is concerned with attacking the nonlinear combining of multiple bits of a single LFSR:

Forre, R. 1989. A Fast Correlation Attack on Nonlinearly Feedforward Filtered Shift-Register Sequences. *Advances in Cryptology -- EUROCRYPT '89.* 586-595. Springer-Verlag.

In the process, Forre discusses (but does not explicitly detail) the original algorithm, and identifies situations where it may fail. The algorithm is then modified for the desired structure, and graphs indicate fairly extensive experimentation with it.

"These experiments showed that the success of the attack depends on the following factors:

- *The number of feedback tabs of the LFSR:* the more taps there are, the more bits are involved in each linear relation and the less reliable is the assignment of probabilities . . . ."
- *The (absolute and relative) heights of the correlation peaks between the running-key sequence and the LFSR-sequence.* Higher peaks are much easier detected by the algorithm than lower ones. . . ."

1990 -- Meier and Staffelbach (Switzerland)

Meier and Staffelbach return with a response to combiners with memory.

Meier, W. and O. Staffelbach. 1990. Correlation Properties of Combiners with Memory in Stream Ciphers. *Advances in Cryptology -- EUROCRYPT '90.* 204-213. Springer-Verlag.

By now it is known that *any* memoryless combiner has a correlation sum equal to one. They say: "the 'total' correlation is independent of the combining function." Then they go on to show a similar result of combining functions with memory (apparently a 1-bit memory).

". . . the summation cipher with two LFSRs can be successfully cryptanalyzed for LFSRs of considerable length with arbitrary feedback connection." "It is shown in [8] that a similar cryptanalysis is no longer possible for a summation cipher with more than 2 LFSRs."

1990 -- Mihaljevic and Golic (Yugoslavia)

Mihaljevic, M. and J. Golic. 1990. A Fast Iterative Algorithm for a Shift Register Initial State Reconstruction Given the Noisy Output Sequence. *Advances in Cryptology -- AUSCRYPT '90.* 165-175. Springer-Verlag.

"This problem of cryptanalysis can be regarded as the problem of a LFSR initial state reconstruction using the noisy output sequence . . . ."

"In this paper we consider a class of algorithms to which Algorithm B [from Meier-Staffelbach] belongs. In this class the initial state reconstruction is based on the error correction principle. It means that the procedure is iterative: in each step we first calculate the posterior probabilities, bit-by-bit (phase I), and them make a bit-by-bit decision (phase II)."

1990 -- Zeng, Yang and Rao (China and USA)

Zeng, K., C. Yang and T. Rao. 1990. An Improved Linear Syndrome Algorithm in Cryptanalysis with Applications. *Advances in Cryptology -- CRYPTO '90.* 34-47.

"What is given is a certain segment of a binary sequence of the form *B = A + X,* where *A* is a linear recursive sequence with known feedback polynomial *f(x)* and the sequence *X* is unknown but sparse in the sense that $Prob(x(t) = 1) = s_0 < 1/2$, s_0 being called the *initial error rate* of the sequence A in the sequence B."

"The method suggests to fix an integer *r >= 3,* find out a set of *r*-nomial multiples . . . of the feedback polynomial *f(x),* compute an odd number, say *2m + 1,* of *syndromes* . . . and revise the signals *b(i)* to new signals *b'(i)* according to the rule of majority decision, namely, put *b'(i) = NOT b(i)* if at least *m + 1* syndromes are 1, otherwise *b'(i) = b(i).*"

"The main idea in the improved LS algorithm is to make the revisions with a **reducing** number of syndromes, with the length of the segment under processing being reduced correspondingly."

The article goes on to give explicit mathematical algorithms for the method. The process is iterated -- repeated -- to improve the "error correction."

1990 -- Staffelbach and Meier (Switzerland)

Staffelbach, O. and W. Meier. 1990. Cryptographic Significance of the Carry for Ciphers Based on Integer Addition. *Advances in Cryptology -- CRYPTO '90.* 601-614.

"Integer addition has been proposed for use in cryptographic transformations since this operation is nonlinear when considered over GF(2)."

"In these ciphers nonlinearity or confusion is achieved via the carry. In fact if the carry happens to be zero, integer addition is linear when considered over *GF(2)* . . . ." "Therefore the strength of these ciphers heavily relies on the randomness of the carry. In particular it is required that the least significant bit (l.s.b.) of the carry is balanced or nearly balanced. However it may happen that this postulate is satisfied in the average, but is violated locally. In fact for the summation combiner with *n* = 2 inputs it has been shown in [4] that the carry is balanced in the average, but is strongly biased in runs of consecutive equal output digits."

"The aim of the present paper is to investigate the probability distribution of the carry for a summation combiner with an arbitrary number *n* of inputs." ". . . it is proved that the carry is balanced for even *n* and biased for odd *n.*"

1990 -- Golic and Mihaljevic (Yugoslavia)

Golic, J. and M. Mihaljevic. 1990. Minimal Linear Equivalent Analysis of a Variable-Memory Binary Sequence Generator. *IEEE Transactions on Information Theory.* 36(1): 190-192.

"Geffe [1] proposed a nonlinear binary sequence generator (BSG) with two linear feedback shift registers (LFSR's) and a memory: one LFSR is used to load the memory from which a binary sequence is read out according to the addresses taken from the other LFSR . . . ." "A somewhat similar principle due to MacLaren and Marsaglia, called randomizing by shuffling, was described in [2] as a way of improving on the statistical properties of random or pseudorandom sequences."

"We consider a BSG consisting of three LFSR's and a memory . . . ." "First, the output bit *b(t)* is read out of the memory location addressed by the read address *X(t)* [from $LFSR_2$]. Second, the output bit *a(t)* of $LFSR_1$ is written into the memory location addressed by the write address *Y(t)* [from $LFSR_3$]. The BSG just described will be referred to as a MEM-BSG. It realizes a time-varying nonlinear function of the phase shifts of a maximum-length sequence."

"It is proved that the linear complexity and the period of output sequences of MEM-BSG are, respectively, at least equal to the linear complexity and the period of output sequences of the corresponding multiplexer-based nonlinear generator, due to Jennings [3] (the MUX-BSG), which consists of two LFSR's and a multiplexer. Moreover, the hardware implementation of the MEM-BSG usually is much simpler than that of the corresponding MUX-BSG."

"Of special interest for spread-spectrum communication systems are the so-called bent-function sequences [9], [14], [15], which possess asymptotically optimal correlation properties." ". . . both the MEM-BSG and the bent-function BSG are suitable for generating fast binary sequences of sufficiently high linear complexities . . . ."

1990 -- Ritter (U.S.A.)

In a little-noticed article, I took combiner design in the other direction: Although previous combiners were concerned only with combining *confusion* (e.g., LFSR) sequences, I was concerned with the data-confusion combiner, because this was the outermost line of defense. Although this required that the design be potentially reversible, a non-reversible version could be used to combine confusion.

Ritter, T. 1990. Substitution Cipher with Pseudo-Random Shuffling: The Dynamic Substitution Combiner. *Cryptologia.* 14(4): 289-303.

"ABSTRACT: A cipher mechanism or process which can be viewed as a modified substitution cipher. A translation table is used to replace plaintext symbols with ciphertext symbols; the modification consists of changing the contents of the translation table after each substitution. The dynamic translation table acts to confuse symbol frequency statistics and so frustrate the usual cryptanalytic attacks."

1991 -- Chepyzhov and Smeets (USSR / Sweden)

Chepyzhov, V. and B. Smeets. 1991. On A Fast Correlation Attack on Certain Stream Ciphers. *Advances in Cryptology -- EUROCRYPT '91.* 176-185. Springer-Verlag.

"**Abstract**--In this paper we present a new algorithm for the recovery of the initial state of a linear feedback shift register when a noisy output sequence is given. Our work is focussed on the investigation of the asymptotical behaviour of the recovery process rather than on the construction of an optimal recovery procedure."

"In the correlation attack as it was originally described by Siegenthaler [1], one uses an exhaustive search through the state space of the shift register. Such a search is not very realistic when the degree *r* (= length of the LFSR) of the feedback polynomial of the LFSR exceeds 60 . . . ." "Recently it was shown by Meier and Staffelbach [2] *[not reviewed here]* that in certain cases one can avoid this exhaustive search. In particular, they showed that if the number *t* of feedback taps is small, then it is possible to restore the initial state by an iterative procedure with much less complexity than exhaustive search."

"Our algorithm is using the key stream almost as efficiently as possible at the expense of an increase of the complexity of the first stage. Our algorithm that we use for the first stage is derived from efficient algorithms for finding the non-zero codeword of lowest weight in a linear code [4], [5]. The second stage of our algorithm is almost identical to Gallager's algorithm for the decoding of low-density parity-check codes [6]."

1991 -- Camion *et. al.* (France)

Here, Camion and others "establish the link between *correlation-immune* functions and *orthogonal arrays."*

Camion, P., C. Carlet, P. Charpin and N. Sendrier. 1991. On Correlation-immune functions. *Advances in Cryptology -- CRYPTO '91.* 86-100. Springer-Verlag.

"**Definition 3.1** An M x m matrix V with entries from a set of q elements is called an orthogonal array of size M, with m constraints, q levels, strength k, and also index u if any set of k columns of V contains all q^k possible row vectors exactly u times. Such an array is denoted by (M,m,q,k). Clearly $M = uq^k$."

"**Theorem 3.1** A boolean function f on G is correlation immune of order k if and only if its truth table is an orthogonal array (M,m,2,k)."

1991 -- Mihaljevic and Golic (Yugoslavia)>

Mihaljevic, M. and J. Golic. 1991. A Comparison of Cryptanalytic Principles Based on Iterative Error-Correction. *Advances in Cryptology -- EUROCRYPT '91.* 527-531. Springer-Verlag.

"**ABSTRACT:** A cryptanalytic problem of a linear feedback shift register initial state reconstruction using a noise output sequence is considered . . . ."

"The following three principles are considered:

"P.1: Error-correction is based on the number of satisfied parity checks.

"P.2: Error-correction is based on the estimation of the relevant posterior probabilities obtained by using the average posterior probability estimated in the previous iteration as the prior probability in the current iteration.

"P.3: Error-correction is based on the estimation of the relevant posterior probabilities obtained by using the posterior probabilities estimated in the previous iteration as the prior probabilities in the current iteration."

Experiments indicate that P.1 is most efficient, and P.3 is somewhat more capable.

1991 -- Zeng, Wang, Wei and Rao (U.S.A.)

Zeng, K., C. Yang, D. Wei and T. Rao. 1991. Pseudorandom Bit Generators in Stream-Cipher Cryptography. *IEEE Computer.* February. 8-17.

"The central problem in stream-cipher cryptography . . . is the difficulty of generating a long unpredictable sequence of binary signals from a short and random key." "The problem is this: On which basis can one draw the conclusion that the output signals of a certain given keystream generator are hard to predict? No universally applicable and practically checkable criteria have been developed to certify the security of bit generators. For that matter, no general theory of cryptanalysis is known to exist except for an ever-expanding arsenal of concrete attack methods elaborated for various practical purposes."

1992 -- Mihaljevic and Golic (Yugoslavia)

Mihaljevic, M. and J. Golic. 1992. Convergence of a Bayesian Iterative Error-Correction Procedure on a Noisy Shift Register. *Advances in Cryptology -- EUROCRYPT '92.* 124-137. Springer-Verlag.

"**ABSTRACT:** Convergence of an algorithm for a linear feedback shift register initial state reconstruction using the noisy output sequence, based on a bitwise Bayesian iterative error-correction procedure, and different weight parity-checks, is analyzed. ..."

"Many of the published keystream generators are based on binary linear feedback shift registers (LFSRs) combined by a memoryless function. Such a generator is called a combination generator."

"In this paper, we consider an iterative algorithm employing the parity-checks of different weights and Bayesian decision rule in error-correction for each bit, assuming that the error-rate from the previous iteration is used as the noise probability in the current one."

1994 -- MacKay (U.K.)

MacKay, D. 1994. A Free Energy Minimization Framework for Inference Problems in Modulo 2 Arithmetic. *Fast Software Encryption.* 179-195. Springer-Verlag.

"**ABSTRACT.** This paper studies the task of inferring a binary vector s given noisy observations of the binary vector t = As modulo 2, where A is a M x N binary matrix." "The unknown binary vector is replaced by a real vector of probabilities that are optimized by variational free energy minimization."

"Consider three binary vectors: **s** [signal] . . . and **r** [received] and **n** [noise] . . . related by:

$$(\mathbf{As} + \mathbf{n}) \bmod 2 = \mathbf{r}$$

where **A** is a binary matrix. Our task is to infer **s** given **r** and **A**, and given assumptions about the statistical properties of **s** and **n**."

"One way to attack a discrete combinatorial problem is to create a related problem in which the discrete variables **s** are replaced by real variables, over which a continuous optimization can then be performed." "In the present context, the question is then 'how should one generalize the posterior probability (4) to the case where **s** is replaced by a vector with real components?'"

"The variational free energy minimization method (Feynman 1972) takes an 'awkward' probability distribution such as the one in (4), and attempts to approximate it by a simpler distribution. . . ."

[MacKay also includes a detailed description of the algorithms in C-like pseudo-code.]

1994 -- Mihaljevic (Yugoslavia)

Mihaljevic, M. 1994, A Correlation Attack on the Binary Sequence Generators with Time-Varying Output Function. *Advances in Cryptology -- ASIACRYPT '94.* 67-79. Springer-Verlag.

"*Abstract:* A binary sequence generator (BSG) consisting of three regularly clocked linear feedback shift registers combined by a time-varying memoryless function is cryptanalysed. A novel distance measure for the binary sequences comparison relevant for the cryptanalysis is proposed . . . ." "It is pointed out that the novel distance based approach to cryptanalysis could be applied for attacking the binary MacLaren-Marsaglia shuffler . . . ."

"The problem is to find out the conditions under which it is possible to reconstruct the initial contents of individual shift registers knowing a segment of the keystream sequence, based on the correlation / statistical dependence between the keystream sequence and a set of the shift register sequences."

"The main objective of this paper is cryptanalysis of a BSG presented in [Golic 1990] which is an extension of similar structures from [Geffe 1973] (the BSG consisting of two LFSR's and a variable memory), [MacLaren and Marsaglia 1968], [Knuth II]. This generator consists of three regularly clocked linear feedback shift registers (LFSR's) combined by a time-varying memoryless function."

1994 -- Menicocci (Italy)

Menicocci, R. 1994, Intrinsic weakness of variable-memory keystream generators. *Electronics Letters.* 30(11): 850-851.

"*Introduction:* The variable-memory binary sequence generator (MEM-BSG) [1] consists of three linear feedback shift registers (LFSRs) and a variable memory. Because of its convenience for generating fast sequences of large period and complexity [1], the MEM-BSG appears suitable for cryptographic applications. In this Letter we point out that there exists a correlation between the output sequence of the generator and the sequence generated by one of the registers. We also show why this correlation represents an intrinsic weakness of the MEM-BSG when used as a keystream generator."

1994 -- Golic (Yugoslavia)

Golic, J. 1994. Intrinsic Statistical Weakness of Keystream Generators. *Advances in Cryptology -- ASIACRYPT '94.* 91-103. Springer-Verlag.

"**Abstract:** It is shown that an arbitrary binary keystream generator with *M* bits of memory can be linearly modelled as a non-autonomous linear feedback shift register of length at most *M* with an additive input sequence of nonbalanced identically distributed binary random variables." "Linear models for clock-controlled shift registers and arbitrary shift register based keystream generators are derived. Several examples including the time-variant memoryless combiner, the basic summation generator, the stop-and-go cascade, and the shrinking generator are presented."

1995 -- Golic *et. al.* (Australia)

Golic, J., M. Salmansizadeh, A. Clark, A. Khodkar and E. Dawson. 1995. Discrete Optimization and Fast Correlation Attacks. *Cryptography: Policy and Algorithms.* 186-200. Springer-Verlag.

"Stream ciphers which generate pseudo-random sequences using the output of a number of linear feedback shift registers (LFSRs) combined by some nonlinear function, with or without memory, have long been proposed for use in secure communications. The purpose of nonlinear combiners is to produce a system which can withstand any practical cryptanalytic attack based on low linear complexity of the observed keystream sequence (see [13]) or high linear correlation to individual LFSR sequences (see [14] and [5])."

"This paper considers the immunity of these combiners to fast divide and conquer correlation attacks [9]. The problem is to find the conditions under which it is possible to reconstruct the initial contents of individual shift registers using a segment of the keystream generator output sequence. Correlation attacks are based on the statistical dependence between the observed keystream sequence and a set of shift register sequences [5], [14]. If such an attack outperforms an exhaustive search over the initial contents of the corresponding shift registers, it is then called a *fast correlation attack."*

1995 -- Klapper and Goresky (U.S.A.)

Klapper, A. and M. Goresky. 1995. Cryptanalysis Based on 2-Adic Rational Approximation. *Advances in Cryptology -- CRYPTO '95.* 262-273.

"The development of cryptosystems tends to alternate between the design of new systems that resist known attacks, and the design of new attacks against systems." "Occasionally a very general attack is found that can potentially be used against a large class of cryptosystems."

"This approach can be used to attack Massey and Rueppel's summation combiner [16, 19]. In their setup, the outputs from several short maximal period LFSRs . . . with pairwise relatively prime periods are combined using addition with carry."

"However, addition with carry is precisely addition in the 2-adic numbers." ". . . if we combine m-sequences of period $2^n - 1$ for *n* = 7, 11, 13, 15, 16, 17, then the resulting sequence has linear span nearly 2^{79}, but the 2-adic span is less than 2^{18}. Thus 2^{19} bits suffice to determine this sequence -- *and far fewer unless care is taken in the choice of m-sequences.*

Terry Ritter, his current address, and his top page.

Last updated: 1996-08-15

Differential Cryptanalysis: A Literature Survey

Research Comments from Ciphers By Ritter

Terry Ritter

Differential Cryptanalysis covers a growing variety of attacks on various block ciphers. It appears to be most useful on iterative (round-based) ciphers, perhaps because these can only weakly diffuse the transformations which occur in later rounds. Differential Cryptanalysis is normally a defined-plaintext attack.

The basic idea of Differential Cryptanalysis is to first cipher some plaintext, then make particular changes in that plaintext and cipher it again. Particular ciphertext differences occur more frequently with some key values than others, so when those differences occur, particular keys are (weakly) indicated. With huge numbers of tests, false indications will be distributed randomly, but true indications always point at the same key values and so will eventually rise above the noise to indicate some part of the key.

The basic concept can be applied to virtually any sort of statistic which relates ciphertext changes to key values, even in relatively weak ways. But because the probabilities involved are generally quite small, success generally depends upon having very substantial amounts of known-plaintext. Thus, in practice, Differential Cryptanalysis would seem to be defeated by the simple use of message keys and limitations on the amount of material ciphered under a single message key.

Some versions (e.g., Biham and Shamir 92) can be applied to separately-keyed blocks with a similar overall probability of success. But that success reveals only *one* of the many keys at random, and a success does not help with the other keys. Nor does Differential Cryptanalysis apply to message keys, since the message key value is not available as known-plaintext. Thus, while Differential Cryptanalysis is *powerful*, it is not *magic*, and has very significant requirements which may not be met in practice.

Differential Cryptanalysis depends upon known tables in which the key value selects various data differentials. Consequently, Differential Cryptanalysis might also be defeated by:

1. Keying which selects among "every" possible table (instead of using a few pre-defined "ideal" tables);
2. Using data to dynamically select among a large working set of tables (instead of just four); and
3. Effectively mixing table results as soon as table operations occur (rather than depending upon future "rounds" for mixing, which is risky since there *are* no future rounds after the last one). Effective mixing should prevent tables from being isolated and separately attacked.

Contents

- 1990
  - Biham and Shamir introduce the concept of Differential Cryptanalysis
- 1991
  - Biham and Shamir take the opportunity to break a variety of ciphers
  - Nyberg gives us "perfect nonlinearity" and a construction for such S-boxes.
  - Dawson and Tavares give us a new set of S-box design criteria based on information theory.
- 1992
  - Biham and Shamir attack "the Full 16-round DES"
  - Nyberg and Knudson give a limit for the size of the differential needed for a successful attack
  - Adams proposes to use bent functions in S-boxes.
- 1993
  - Ben-Aroya and Biham attack Lucifer
  - O'Connor examines the expected Differential Cryptanalysis effects of random S-box selection.
- 1995
  - Youssef, Tavares, Mister and Adams gives "the expected nonlinearity of a randomly selected injective substitution box."
  - Youssef and Tavares discusses the immunity of randomly selected S-boxes to differential cryptanalysis and linear cryptanalysis.
  - Vaudenay says that S-box linearity is not so important.

1990 -- Biham and Shamir

Biham, E. and A. Shamir. 1990. Differential Cryptanalysis of DES-like Cryptosystems. *Advances in Cryptology -- CRYPTO '90.* Springer-Verlag. 2-21.

"*Iterated cryptosystems* are a family of cryptographically strong functions based on iterating a weaker function *n* times. Each iteration is called a *round* and the cryptosystem is called an *n round cryptosystem.* The *round function* is a function of the output of the previous round and of a *subkey* which is a key dependent value calculated via a *key scheduling* algorithm. The round function is usually based on S boxes, bit permutations, arithmetic operations and the exclusive-or (denoted by + and XOR) operations. The *S boxes* are nonlinear translation tables mapping a small number of input bits to a small number of output bits. They are usually the only part of the cryptosystem which is not linear and thus the security of the cryptosystem crucially depends upon their choice. The bit permutation is used to rearrange the output bits of the S boxes in order to make the input bits of each S box in the following round depend upon the output of as many S boxes as possible."

"In this paper we describe a new kind of attack that can be applied to many DES-like iterated cryptosystems. This is a chosen plaintext attack which uses only the resultant ciphertexts. The basic tool of the attack is the *ciphertext pair* which is a pair of ciphertexts whose plaintexts have particular differences. The two plaintexts can be chosen at random, as long as they satisfy the difference condition, and the cryptanalyst does not have to know their values. The attack is statistical in nature and can fail in rare instances."

"*Differential cryptanalysis* is a method which analyzes the effect of particular differences in plaintext pairs on the differences of the resultant ciphertext pairs. These differences can be used to assign probabilities to the possible keys and to locate the most probable key. This method usually works on many pairs of plaintexts with the same particular difference using only the resultant ciphertext pairs. For DES-like cryptosystems the difference is chosen as a fixed XORed value of two plaintexts."

"Although DES seems to be very non linear in its input bits, when particular combinations of input bits are modified simultaneously, particular intermediate bits are modified in a usable way with a relatively high probability after several rounds."

". . . every input XOR of an S box suggests a probabilistic distribution of the possible output XORs. In this distribution several output XORs have a relatively high probability. Table 1 describes the distribution of the output XORs for several input XORs in S1." "Many entries are impossible or have a small probability. However, there are several entries with probability 1/4 (i.e., 16 out of 64) or close to it."

"We can use this property as a tool to identify key bits. If we can find the output XOR of the *F* function of the last round, we can filter the set of possible subkeys entering this *F* function when the pair of ciphertexts is known. Using both ciphertexts it is easy to calculate the input XOR of the *F* function of the last round and its output XOR. Then the input XOR and output XOR of each S box in the last round are known. In case *k* input pairs can lead to that entry in the table, exactly *k* values of the corresponding six subkey bits are possible. Most subkey values are suggested by only a few pairs. However, the real value of the subkey bits is suggested by all the pairs and can be identified."

1991 -- Biham and Shamir

Biham, E. and A. Shamir. 1991. Differential Cryptanalysis of Snefru, Khafre, REDOC-II, LOKI and Lucifer. *Advances in Cryptology -- CRYPTO '91.* Springer-Verlag. 156-171.

"Two-pass Snefru is easily breakable within three minutes on a personal computer."

"Khafre with 16 rounds is breakable by a differential cryptanalytic chosen plaintext attack using about 1500 encryptions within about an hour on a personal computer."

"REDOC-II with one round is breakable by a differential cryptanalytic chosen plaintext attack using about 2300 encryptions within less than a minute on a personal computer."

1991 -- Nyberg

Nyberg, K. 1991. Perfect nonlinear S-boxes. *Advances in Cryptology -- EUROCRYPT '91.* 378-386.

Abstract

"A perfect nonlinear S-box is a substitution transformation with evenly distributed directional derivatives. Since the method of differential cryptanalysis presented by E. Biham and A. Shamir makes use of nonbalanced direction derivatives, the perfect nonlinear S-boxes are immune to this attack. The main result is that for a perfect nonlinear S-box the number of input variables is at least twice the number of output variables." [p.378]

1. Introduction

"In [12] Meier and Staffelbach discuss perfect nonlinear Boolean functions, which are defined to be at maximum distance from linear structures. These functions are the same as the previously known bent functions [15]. To construct perfect nonlinear S-boxes it is necessary that each output bit is a perfect nonlinear function of the input. But it is not sufficient, indeed, also every linear combination of output variables have to be perfect nonlinear. We present two different constructions to achieve this property." [p.378]

1991 -- Dawson and Tavares

Dawson, M. and S. Tavares. 1991. An Expanded Set of S-box Design Criteria Based on Information Theory and its Relation to Differential-Like Attacks. *Advances in Cryptology -- EUROCRYPT '91.* 353-367.

Introduction

"In this work we present an expanded set of design criteria for creating good S-boxes based on information theoretic concepts and show that an S-Box that meets these criteria is immune to differential cryptanalysis [1]."

"We have defined a set of six properties that an Ideal S-box is required to meet. This set of properties has a broader scope than those of Forre and any S-box that meets these properties will also meet Forre's. The properties are grouped into a set of static properties and a set of dynamic properties."

Static Properties

"The first static property is that the partial information about the inputs and outputs does not reduce the uncertainty in an unknown output."

"The second static property is that the partial information about the inputs and outputs does not reduce the uncertainty in an unknown output."

"The third static property is that the uncertainty in a data value is reduced by the minimum amount possible when it passes through an S-box."

Dynamic Properties

"The dynamic properties are similar to the static properties except that they deal with the changes in inputs and outputs."

Analysis of DES S-boxes Using The Design Criteria

". . . we could not find S-boxes with substantially better information theoretic properties than the S-boxes of DES and which also meet the acknowledged DES design criteria." ". . . there were many S-boxes found which met the acknowledged DES design criteria but had poor information theoretic properties."

". . . the properties of the inverses of the DES 4x4 S-boxes were as good as those of the S-boxes themselves." ". . . the inverses of the DES 4x4 S-boxes meet the acknowledged DES design criterion which requires that at least two bits change in the output whenever one input bit is changed. These two discoveries indicate that the designers of DES placed an equal emphasis on the properties of the S-boxes and their inverses.

"In every case we found that the properties of the complete 6x4 S-boxes were better than any individual 4x4 sub-box. We concluded that using multiple sub-boxes to form a larger S-box is an important method which can be used to create S-boxes that have better properties than are possible in a single S-box."

". . . no *nxn* S-box can meet the Dynamic criteria perfectly because, due to the nature of the XOR function, output XOR values always occur in pairs (since a XOR b = b XOR a)."

1992 -- Biham and Shamir

Biham, E. and A. Shamir. 1992. Differential Cryptanalysis of the Full 16-round DES. *Advances in Cryptology -- CRYPTO '92.* Springer-Verlag. 487-496.

"In this paper we finally break the 16-round barrier. We develop an improved version of differential cryptanalysis which can break the full 16-round DES in 2^{37} time and negligible space by analyzing 2^{36} ciphertexts obtained from a larger pool of 2^{47} chosen plaintexts. An interesting feature of the new attack is that it can be applied with the same complexity and success probability even if the key is frequently changed and thus the collected ciphertexts are derived from many different keys."

"Any pair of plaintexts which gives rise to the intermediate XORs specified by this characteristic is called a right pair. The attack tries many pairs of plaintexts, and eliminates any pair which is obviously wrong due to its known input and output values. However since the cryptanalyst cannot actually determine the intermediate values, the elimination process is imperfect and leaves behind a mixture of right and wrong pairs."

"In earlier versions of differential cryptanalysis, each surviving pair suggested several possible values for certain key bits. Right pairs always suggest the correct value for these key bits (along with several wrong values), while wrong pairs suggest random values. When sufficiently many right pairs are analyzed, the correct value (signal) overcomes the random values (noise) by becoming the most frequently suggested value. The actual algorithm is to keep a separate counter for the number of times each value is suggested, and to output the index of the counter with the maximal final value. This approach requires a huge memory (with up to 2^{42} counters in the attack on the 15-round variant of DES), and has a negligible probability of success when the number of analyzed pairs is reduced below the threshold implied by the signal to noise ratio.

"In the new version of differential cryptanalysis, we work somewhat harder on each pair, and suggest a list of complete 56-bit keys rather than possible values for a subset of key bits. As a result, we can immediately test each suggested key via trial encryption, without using any counters. These tests can be carried out in parallel on disconnected processors with very small local memories, and the algorithm is guaranteed to discover the correct key as soon as the first right pair is encountered. Since the processing of different pairs are unrelated, they can be generated by different keys at different times due to frequent key changes, and the discovery of a key can be announced in real time while it is still valid (e.g., in order to forge authenticators for banking messages)."

1992 -- Nyberg and Knudsen

Nyberg, K. and L. Knudsen. 1993. Provable Security Against Differential Cryptanalysis. *Advances in Cryptology -- CRYPTO '92.* 566-574.

1 Introduction

"The purpose of this paper is to show that there exist DES-like iterated ciphers which are provably resistant against differential attacks."

2 Differential Cryptanalysis of DES-like Iterated Ciphers

"In [1] Biham and Shamir introduced differential cryptanalysis of DES-like ciphers. In their attacks they make use of characteristics, which describe the behaviour of input and output differences for some number of consecutive rounds. The probability of a one-round characteristic is the conditional probability that given a certain difference in the inputs to the round we get a certain difference in the outputs of that round. Assume that in every round the inputs *E*(*R*) XOR *K* to **f** are independent and random. This assumption is satisfied if the round keys are uniformly random and independent. Then the probability of an *r*-round characteristic is obtained by multiplying the probabilities of the *r* one-round characteristics.

"Lai and Massey [3] observed that for the success of differential cryptanalysis it is not necessary to fix the values of input and output differences for the intermediate rounds in a characteristic. They introduced the notion of *differentials*. The probability of an *r*-round differential is the conditional probability that given an input difference at the first round, the output difference at the *r*'th round will be some fixed value. Note that the probability of an *r*-round differential with input difference *A* and output difference *B* is the sum of the probabilities of all *r*-round characteristics with input difference *A* and output difference *B*. For *r* <= 2 the probabilities for a differential and for the corresponding characteristic are equal, but in general the probabilities for differentials would be higher.

"In order to make a successful attack on a DES-like iterated cipher by differential cryptanalysis the existence of good characteristics is sufficient. On the other hand to prove security against differential attacks for DES-like iterated ciphers we must ensure that there is no differential with a probability high enough to enable successful attacks."

1992 -- Adams

Adams, C. 1992. On immunity against Biham and Shamir's "differential cryptanalysis." *Information Processing Letters.* 41: 77-80.

2. Avoiding differential cryptanalysis

"Differential cryptanalysis [5] is based on the fact that in many s-boxes certain input XORs (i.e., certain fixed changes in the s-box input vector) lead to certain output XORs (fixed changes in the s-box output vector) with fairly high probability ([about] 25%) and to certain other output XORs with very low or zero probability. Chosen plaintext attacks can be mounted which take advantage of the relatively high probabilities to reduce the search space for the key in use. It is obvious, therefore, that if all output XORs occurred with similar (ideally, equal) probability, differential cryptanalysis would have no greater chance of success than exhaustive search.

"We can design s-boxes with equiprobable output XORs through the use of bent functions ([10,14,2], and others)."

". . . the s-boxes described above cannot be *n x n* bijective s-boxes since columns in the representative matrix are bent and bent functions are not weight balanced. Therefore, SPN cryptosystems taking advantage of this work must be constructed such that it is never required to go 'backwards' through any of their component s-boxes."

1993 -- Ben-Aroya and Biham

Ben-Aroya, I. and E. Biham. 1993. Differential Cryptanalysis of Lucifer. *Advances in Cryptology -- CRYPTO '93.* Springer-Verlag. 186-199.

Abstract

"Differential cryptanalysis was introduced as an approach to analyze the security of DES-like cryptosystems. The first example of a DES-like cryptosystem was Lucifer, the direct predecessor of DES, which is still believed by many people to be much more secure than DES, since it has 128 key bits, and since no attacks against (the full variant of) Lucifer were ever reported in the cryptographic literature. In this paper we introduce a new extension of differential cryptanalysis, devised to extend the class of vulnerable cryptosystems. This new extension suggests key-dependent characteristics, called *conditional characteristics,* selected to enlarge the characteristics' probabilities for keys in subsets of the key space. The application of conditional characteristics to Lucifer shows that more than half of the keys of Lucifer are insecure, and the attack requires about 2^{36} complexity and chosen plaintexts to find those keys. The same extension can also be used to attack a new variant of DES, called RDES, which was designed to be immune against differential cryptanalysis. These new attacks flash new light on the design of DES, and show that the transition of Lucifer to DES strengthened the later cryptosystem."

"In this paper we extend differential cryptanalysis in several directions: The main extension of this paper lets differential cryptanalysis to analyze a wider set of cryptosystems. We define *conditional characteristics* as key-dependent characteristics selected to maximize the characteristic's probability (the fraction of right pairs) for only a specific subset of the key space. The required coverage of (almost) all the key space is done via selection of several conditional characteristics designed for different fractions of the key space."

"Several researchers studied how to make cryptosystems immune against differential analysis, but till now, this effort was not very successful. Many of them[1,9,18] suggested the use of S boxes whose difference distribution tables are uniform, and in particular they suggested the use of bent functions. However, the application of this suggestion to DES was studied in [2,7], and it was shown that the resultant cryptosystems become much weaker than DES."

"Differential cryptanalysis requires one to find good characteristics, i.e., to find pairs of messages, such that the difference of the output of the *n*th round during encryption of these messages is predictable with a relatively high probability." "In [3,2] the characteristic's probability is defined as the probability that a random pair (whose plaintext difference is $omega_p$) is a right pair with respect to a random key, and it is shown that the probability that a random pair is a right pair with respect to a fixed key may depend on the choice of the key. In this paper we are interested in characteristics for which the probability that a random pair is a right pair vary between different keys. We call these characteristics *conditional characteristics.*"

1993 -- O'Connor

O'Connor, L. 1993. On the Distribution of Characteristics in Bijective Mappings. *Advances in Cryptology -- EUROCRYPT 93.* 360-370.

1 Introduction and Results

"Differential cryptanalysis is a statistical attack popularized by Biham and Shamir in a series of well-known papers [1, 2, 3]. The attack has been applied to a wide range of iterated mappings including LUCIFER, DES, FEAL, REDOC, Kahfre [4, 5, 12, 13, 17, 19]. As explained below, the attack is based on a quantity O called a *characteristic,* which has some probability p^{O} of giving information about the secret key used in the mapping. The attack is universal in that characteristics O will always exist for any iterated mapping; however p^{O} may be very small, and possibly less likely to furnish information concerning the key than the success of guessing the secret key at random. For this reason, differential cryptanalysis has had varying success against the iterated mappings it has been applied to, and little is known about how useful the attack is expected to be against an arbitrary iterated mapping."

"We will give a brief description of differential cryptanalysis with reference to product ciphers, though any iterated mapping would suffice. For a product cipher *E* that consists of *R* rounds, let $E_r(X,K)$ be the encryption of the plaintext *X* under the key *K* for *r* rounds, 1 <= *r* <= *R*. Note that $E_R(X,K) = E(X,K) = C$ is the ciphertext for *X*. Let $dC(r) = E_r(X,K) + E_r(X',K)$ be the difference between the ciphertexts of two plaintexts *X,X'* after *r* rounds where 1 <= *r* <= *R*. For our purposes the difference operator + will refer to addition in the vector space Z_2^m. An *r*-round *characteristic* is defined as an (*r*+1)-tuple $O_R(dX,dY_1, dY_2,...,dY_r)$ where *dX* is a plaintext difference, and the dY_i are ciphertext differences. A plaintext pair *X,X'* of difference $dX = X + X'$ is called a *right pair* with respect to a key *K* and a characteristic $O_r(dX,dY_1, dY_2,...,dY_r)$ if when *X* and *X'* are encrypted, $dC(i) = dY_i$ for 1 <= *i* <= *r*. That is, *X,X'* is a right pair if the characteristic correctly predicts the ciphertext differences at each round. The characteristic O_r has probability p^{O_r} if a fraction p^{O_r} of the plaintext pairs of difference *dX* are right pairs. On the other hand, if *X,X'* such that $dX = X + X'$ is not a right pair, then it is said to be a *wrong pair* (with respect to the characteristic and the key). A table which records the number of pairs of difference *dX* that give the output difference *dY* for a mapping PI is called the *XOR table distribution of PI*. A characteristic *dX,dY* is said to be *impossible* for PI if its corresponding XOR table entry is zero. Also a characteristic will be called *nonzero* if $w(dX),w(dY) > 0$, where *w*(.) is the Hamming weight function. Using a characteristic O of appropriate length it is then possible to devise a statistical experiment which when repeated a sufficient number of times will yield the subkey of the last round (see [1] for details)."

4 Conclusion and Remarks

"Our results then show that a relatively simple design can produce product ciphers for which all characteristics O are expected to (correctly) predict differences with low probability. We further note that random *m*-bit permutations can be generated efficiently [15], and that the fraction of permutations that are . . . linear [7] or degenerate [14] in any output bit is tending to zero rapidly as a function of *m*. On the other hand, Biham and Shamir [3] found that replacing the S-boxes of DES by random 4-bit permutations yielded systems that were far weaker than the original DES. The weakness of these S-boxes appears to be due to the dimension of the permutation rather than the use of [random] permutations *per se.*"

1995 -- Youssef, Tavares, Mister and Adams

Youssef, A., S. Tavares, S. Mister and C. Adams. 1995. Linear Approximation of Injective S-boxes. *IEE Electronics Letters.* 31(25): 2168-2169.

Abstract

"Nonlinearity is a crucial requirement for the substitution boxes in secure block ciphers. In this letter, we derive an estimate for the expected nonlinearity of a randomly selected injective substitution box."

Introduction

"Differential cryptanalysis [1] and linear cryptanalysis [2] are powerful cryptanalytic attacks on private-key block ciphers. The complexity of differential cryptanalysis depends on the size of the largest entry in the XOR table, the total number of zeros in the XOR table, and the number of nonzero entries in the first column of that table [1], [3]. The complexity of linear cryptanalysis depends on the size of the largest entry in the linear approximation table (LAT)[2].

"One way to reduce the size of the largest entry in the XOR table is to use injective substitution boxes (s-boxes) such that the number of output bits of the s-box is sufficiently larger than the number of input bits. In this way, it is very likely that the entries in the XOR distribution table of a randomly chosen injective s-box will have only small values, making the block cipher resistant to differential cryptanalysis. Some proposed block ciphers, such as CAST [4] and Blowfish [5], take advantage of this property.

"On the other hand, Biham [6] proved that if for an *nxm* s-box described by $f: Z_2^n \to Z_2^m$ we have $m >= 2^n - n$, then at least one linear combination of the output bits must be an affine combination of the input bits and the block cipher can be trivially broken by linear cryptanalysis. In this letter, we estimate the size of the largest entry in the LAT of a randomly selected injective s-box."

1995 -- Youssef and Tavares

Youssef, A., S. Tavares. 1995. Resistance of Balanced S-boxes to Linear and Differential Cryptanalysis. *Information Processing Letters.* 56: 249-252.

Abstract

"In this letter, we study the marginal density of the XOR distribution table, and the linear approximation table entries of regular substitution boxes (s-boxes). Based on this, we show that the fraction of good s-boxes (with regard to immunity against linear and differential cryptanalysis) increases dramatically with the number of input variables."

Introduction

"Differential cryptanalysis [1], and linear cryptanalysis [3] are currently the most powerful cryptanalytic attacks on private-key block ciphers. The complexity of differential cryptanalysis depends on the size of the largest entry in the XOR table, the total number of zeros in the XOR table, and the number of nonzero entries in the first column in that table [1], [8]. The complexity of linear cryptanalysis depends upon the size of the largest entry in the linear approximation table (LAT).

"One requirement in s-box design is to have a balanced s-box (also known as a regular s-box). This means that each output symbol should appear an equal number of times when the input is varied aver all possible values.

"Gordon and Retkin calculated the probability that one or more of the output coordinates of a random, reversible s-box is an affine function. By showing that this probability decreases dramatically with the number of input variables, they conjectured that larger s-boxes are better. In this letter, we provide further evidence for their conjecture by showing that the fraction of good s-boxes, with regard to immunity against linear and differential cryptanalysis, increases dramatically with the number of input variables."

1995 -- Vaudenay

Vaudenay, S. 1995. An Experiment on DES Statistical Cryptanalysis.

Abstract

"Linear cryptanalysis and differential cryptanalysis are the most important methods of attack against block ciphers. Their efficiency have been demonstrated against several ciphers, including the Data Encryption Standard. We prove that both of them can be considered, improved and joined in a more general statistical framework. We also show that the very same results as those obtained in the case of DES can be found without any linear analysis and we slightly improve them into an attack with theoretical complexity $2^{42.9}$.

"We can apply another statistical attack -- the X^2-cryptanalysis -- on the same characteristics without a definite idea of what happens in the encryption process. It appears to be roughly as efficient as both differential and linear cryptanalysis."

"The success of those methods have focused the attention on the linear properties of the boxes. In this paper, we try to prove that the linear properties are not so important."

Terry Ritter, his current address, and his top page.

Last updated: 1997-09-25

Linear Cryptanalysis: A Literature Survey

Research Comments from Ciphers By Ritter

Terry Ritter

Perhaps the best introduction to Linear Cryptanalysis is the original article by Matsui, although continued development can be expected to have changed this original approach somewhat.

Apparently, Linear Cryptanalysis starts by finding approximate linear expressions for S-boxes, then extends these expressions to the entire cipher. Clearly, if the expressions were precisely linear, known-plaintext could immediately be "solved" for key bits.

As I understand it, since the expressions are only approximate, in each expression a particular value for a key bit may only be slightly more probable than its complement. Accordingly, considerable known-plaintext is required before key bit values are clearly indicated.

The question for new cipher designs is whether we can ever prove that no approximate linear expression exists which is sufficiently effective as to expose the key. One answer to this is to key the S-boxes, thus depriving the analyst of precise knowledge of their contents which means that they cannot be reasonably approximated.

Contents

- 1993
  - Matsui introduces the linear cryptanalysis of DES.
- 1994
  - Matsui and Yamagishi deal with FEAL.
  - Matsui gives an actual experimental cryptanalysis of DES.
  - Daemen, Govaerts and Vandewalle introduce "the correlation matrix of a Boolean mapping" which is said to be "the 'natural' representation for the proper understanding and description of the mechanisms of linear cryptanalysis."
  - Kaliski and Robshaw give a form of linear cryptanalysis using multiple linear approximations.
- 1995
  - Youssef, Tavares, Mister and Adams gives "the expected nonlinearity of a randomly selected injective substitution box."
  - Youssef and Tavares discusses the immunity of randomly selected S-boxes to differential cryptanalysis and linear cryptanalysis.
  - Vaudenay says that S-box linearity is not so important.
  - Harpes, Kramer and Massey argue that IDEA and SAFER K-64 really are OK.
  - Buttyan and Vajda "show that the problem of searching for the best characteristic in linear cryptanalysis is equivalent to searching for the maximal weight path in a directed graph."

1993 -- Matsui

Matsui, M. 1993. Linear Cryptanalysis Method for DES Cipher. *Advances in Cryptology -- EUROCRYPT* '93. 386-397.

Abstract

"We introduce a new method for cryptanalysis of DES cipher, which is essentially a known-plaintext attack. As a result, it is possible to break 8-round DES cipher with 2^{21} known-plaintexts and 16-round DES cipher with 2^{47} known-plaintexts."

1 Introduction

"In this paper we introduce an essentially known-plaintext attack of DES cipher. The purpose of this method is to obtain a linear approximate expression of a given cipher algorithm. For this purpose, we begin by constructing a statistical linear path between input and output bits of each S-box. Then we extend this path to the entire algorithm, and finally reach a linear approximate expression without any intermediate value."

"Generally speaking, there exist many linear approximate expressions for a given cipher algorithm. Moreover, if plaintexts are not random, we may even find an expression which has no plaintext bits in it. This suggests that our method finally leads to an only-ciphertext attack."

1994 -- Matsui and Yamagishi

Matsui, M. and A. Yamagishi. 1994. A New Cryptanalytic Method for FEAL Cipher. *IEICE Transactions on Fundamentals of Electronics, Communications and Computer Science.* E77-A(1): 2-7.

Introduction

"In this paper, we propose a new technique of a known plaintext attack of FEAL cipher. Our method is a kind of meet-in-the-middle attack with a partial exhaustive key search, and therefore we can derive all possible key candidates directly and deterministicly. In other words, it enables us to specify all candidates for secret key *K* which satisfies *Encryption*(*P*,*K*) = (*C*) for any of given plaintexts *P* and the corresponding ciphertext *C*.

"To realize this, we newly introduce a 'checking function' and a 'cutting off method.' The former is a function *g*(*P*,*C*,*K*) whose value is constant if and only if the key candidate *K* satisfies *Encryption*(*P*,*K*) = (*C*) for any plaintext *P* and the corresponding ciphertext *C*, and the latter is a technique to reduce the number of key candidates *K*."

1994 -- Matsui

Matsui, M. 1994. The First Experimental Cryptanalysis of the Data Encryption Standard. *Advances in Cryptology -- CRYPTO* '94. 1-11.

Introduction

"In the first paper on linear cryptanalysis [2], we introduced a new measure of linearity of S-boxes and extended it to the entire cipher structure of DES."

"This paper studies an improved version of linear cryptanalysis and its application to the first successful computer experiment in breaking the full 16-round DES. We newly introduce two viewpoints; linear approximate equations based on the best (*n*-2)-round expression, and reliability of the key candidates derived from these equations. The former reduces the number of required plaintexts, whereas the latter increases the success rate of our attack.

In the 2^{47}-method, we established two linear approximate equations of 16-round DES using the best 15-round expression, where each equation includes one active S-box and hence recovers 7 secret key bits. This paper, however, begins with two new linear approximate equations derived from the best 14-round expression, where each equation has two active S-boxes and can recover 13 secret key bits. These equations give us, therefore, a total of 26 secret key bits, and then the remaining 56 - 26 = 30 secret key bits are within the reach of an exhaustive search."

"As a result, DES is breakable with complexity 2^{43} and success rate 85% if 2^{43} known-plaintexts are available. For another example, success rate is 10% with complexity 2^{50} if 2^{38} known-plaintexts are available.

"We carried out the first experimental attack of the full 16-round DES using twelve computers (HP9735/PA-RISK 99MHz) to confirm this scenario. The program, described in C and assembly languages consisting of a total of 1000 lines, was designed to solve two equations while generating 2^{43} random plaintexts and enciphering them. We finally reached all of the 56 secret key bits in fifty days, out of which forty days were spent for generating plaintexts and their ciphertexts and only ten days were spent for the actual key search."

1994 -- Daemen, Govaerts and Vandewalle

Daemen, J., R. Govaerts and J. Vandewalle. 1995. Correlation Matrices. *Fast Software Encryption.* Lecture Notes in Computer Science (LNCS) 1008. Springer-Verlag. 275-285.

Abstract

"In this paper we introduce the *correlation matrix* of a Boolean mapping, a useful concept in demonstrating and proving properties of Boolean functions and mappings. It is argued that correlation matrices are the "natural" representation for the proper understanding and description of the mechanisms of linear cryptanalysis [4]. It is also shown that the difference propagation probabilities and the table consisting of the squared elements of the correlation matrix are linked by a scaled Walsh-Hadamard transform."

1 Introduction

"Most components in encryption schemes are Boolean mappings. In this paper, we establish a relation between Boolean mappings and specific linear mappings over real vector spaces. The matrices consist of the correlation coefficients associated with linear combinations of input bits and linear combinations of output bits."

1994 -- Kaliski and Robshaw

Kaliski, B. and M. Robshaw. 1994. Linear Cryptanalysis Using Multiple Approximations. *Advances in Cryptology -- CRYPTO '94.* 26-39.

1 Introduction

"Matsui and Yamagishi [6] introduced the idea of *linear cryptanalysis* in 1992 on an attack on FEAL [10]. The techniques used in this attack were refined by Matsui and used with dramatic effect on DES [7] in a theoretical attack on the full 16-round DES requiring 2^{47} known plaintext / ciphertext pairs [4]. After further work an experiment was performed during which the key used in the full 16-round version of DES was recovered using 2^{43} known plaintext / ciphertext pairs [9].

"The most notable feature about linear cryptanalysis is that it is a known plaintext attack rather than a chosen plaintext attack (differential cryptanalysis [1] is a chosen plaintext attack) and as such poses more of a practical threat to a block cipher. At present, however, a successful linear cryptanalytic attack on DES still requires a large quantity of known plaintext.

"In this paper we consider an extension to the linear cryptanalytic attack [4, 5] using multiple linear approximations. This offers a slight improvement in the efficiency of an attack on the DES but more importantly, it is generally applicable and in certain circumstances it might well be extremely effective in reducing the amount of data required by a cryptanalyst for a successful attack on a block cipher using linear cryptanalysis."

1995 -- Youssef, Tavares, Mister and Adams

Youssef, A., S. Tavares, S. Mister and C. Adams. 1995. Linear Approximation of Injective S-boxes. *IEE Electronics Letters.* 31(25): 2168-2169.

Abstract

"Nonlinearity is a crucial requirement for the substitution boxes in secure block ciphers. In this letter, we derive an estimate for the expected nonlinearity of a randomly selected injective substitution box."

Introduction

"Diferential cryptanalysis [1] and linear cryptanalysis [2] are powerful cryptanalytic attacks on private-key block ciphers. The complexity of differential cryptanalysis depends on the size of the largest entry in the XOR table, the total number of zeros in the XOR table, and the number of nonzero entries in the first column of that table [1], [3]. The complexity of linear cryptanalysis depends on the size of the largest entry in the linear approximation table (LAT)[2].

"One way to reduce the size of the largest entry in the XOR table is to use injective substitution boxes (s-boxes) such that the number of output bits of the s-box is sufficiently larger than the number of input bits. In this way, it is very likely that the entries in the XOR distribution table of a randomly chosen injective s-box will have only small values, making the block cipher resistant to differential cryptanalysis. Some proposed block ciphers, such as CAST [4] and Blowfish [5], take advantage of this property.

"On the other hand, Biham [6] proved that if for an *nxm* s-box described by f: $Z_2^n \to Z_2^m$ we have $m >= 2^n - n$, then at least one linear combination of the output bits must be an affine combination of the input bits and the block cipher can be trivially broken by linear cryptanalysis. In this letter, we estimate the size of the largest entry in the LAT of a randomly selected injective s-box."

1995 -- Youssef and Tavares

Youssef, A., S. Tavares. 1995. Resistance of Balanced S-boxes to Linear and Differential Cryptanalysis. *Information Processing Letters.* 56: 249-252.

Abstract

"In this letter, we study the marginal density of the XOR distribution table, and the linear approximation table entries of regular substitution boxes (s-boxes). Based on this, we show that the fraction of good s-boxes (with regard to immunity against linear and differential cryptanalysis) increases dramatically with the number of input variables."

Introduction

"Differential cryptanalysis [1], and linear cryptanalysis [3] are currently the most powerful cryptanalytic attacks on private-key block ciphers. The complexity of differential cryptanalysis depends on the size of the largest entry in the XOR table, the total number of zeros in the XOR table, and the number of nonzero entries in the first column in that table [1], [8]. The complexity of linear cryptanalysis depends upon the size of the largest entry in the linear approximation table (LAT).

"One rerquirement in s-box design is to have a balanced s-box (also known as a regular s-box). This means that each output symbol should appear an equal number of times when the input is varied aver all possible values.

"Gordon and Retkin calculated the probability that one or more of the output coordinates of a random, reversible s-box is an affine function. By showing that this probability decreases dramatically with the number of input variables, they conjectured that larger s-boxes are better. In this letter, we provide further evidence for their conjecture by showing that the fraction of good s-boxes, with regard to immunity against linear and differential cryptanalysis, increases dramatically with the number of input variables."

1995 -- Vaudenay

Vaudenay, S. 1995. An Experiment on DES Statistical Cryptanalysis.

Abstract

"Linear cryptanalysis and differential cryptanalysis are the most important methods of attack against block ciphers. Their efficiency have been demonstrated against several ciphers, including the Data Encryption Standard. We prove that both of them can be considered, improved and joined in a more general statistical framework. We also show that the very same results as those obtained in the case of DES can be found without any linear analysis and we slightly improve them into an attack with theoretical complexity $2^{42.9}$.

"We can apply another statistical attack -- the χ^2-cryptanalysis -- on the same characteristics without a definite idea of what happens in the encryption process. It appears to be roughly as efficient as both differential and linear cryptanalysis."

"The success of those methods have focused the attention on the linear properties of the boxes. In this paper, we try to prove that the linear properties are not so important."

1995 -- Harpes, Kramer and Massey

Harpes, C., G. Kramer and J. Massey. 1995. A Generalization of Linear Cryptanalysis and the Applicability of Matsui's Piling-up Lemma. *Advances in Cryptology -- EUROCRYPT* '95. 24-38.

Abstract

"Matsui's linear cryptanalysis for iterated block ciphers is generalized by replacing his linear expressions with I/O sums. For a single round, an I/O sum is the XOR of a balanced binary-valued function of the round input and a balanced binary-valued function of the round output." "A cipher contrived to be secure against linear cryptanalysis but vulnerable to this generalization of linear cryptanalysis is given. Finally, it is argued that the ciphers IDEA and SAFER K-64 are secure against this generalization."

1995 -- Buttyan and Vajda

Buttyan, L. and I Vajda. 1995. Searching for the best linear approximation of DES-like cryptosystems. *Electronics Letters.* 31(11): 873-874.

"The authors show that the problem of searching for the best characteristic in linear cryptanalysis is equivalent to searching for the maximal weight path in a directed graph."

"We implemented the algorithm with type 1 restriction for a DES cryptosystem on a personal computer. We obtained the same best linear expressions as Matsui [2] in a few minutes. If better approximations were required, we should approximate s > 1 S-boxes in round function *F*. Unfortunately, this leads to rapid growth in the graph." "Even for s = 2, the direct implementation of the algorithm needs [about] 10^{12} bytes of memory, which makes it infeasible. This kind of problem means that there is a feasibility limit to such cryptanalysis in general. Suboptimal solutions could be expected by applying sequential decoding type algorithms for searching characteristics better than type 1 optimums."

Terry Ritter, his *current address*, and his *top page*.

Last updated: 1997-09-25

Walsh-Hadamard Transforms: A Literature Survey

Research Comments from Ciphers By Ritter

Terry Ritter

Most introductions to Walsh-Hadamard transforms are heavily mathematical and difficult to relate to practice. But the transformation process itself is just arithmetic, and has an easily-comprehended structure which could be widely understood. The result is often thought of as "a poor-man's fast Fourier transform (FFT)" representing the conversion of a time-sampled signal into an equivalent frequency-sampled form. Since a fast Walsh transform (FWT) is much faster than an FFT, there is ample motive to seek reasonable applications. Unfortunately, the Walsh-Hadamard form of "digital frequency," or *sequency* is not intuitively close to the sine-wave form we normally associate with "frequency."

Fortunately, other aspects of the Walsh-Hadamard representation can be useful on their own. The transform can identify correlations in combining functions. Transformed data can be easier to manipulate for differential or integral equation solution, and some parts of digital circuit synthesis. There is apparently also an inherent relation to linear feedback shift registers which may be worth exploiting.

Contents

- 1969
  - *Algorithm:* Shanks gives an algorithm producing the dyadic (Paley) or bit-reversed form of the result.
- 1970
  - *Comment:* Henderson indicates that the bit-reversed form produced by Shanks can be modified to produce sequency-ordered results.
  - *Comment:* Yuen notes that the "Paley" ordering is related to sequency ordering through a Gray code.
- 1971
  - *Application:* Kak applies Walsh-Fourier transforms to measuring the amount of randomness in a finite random sequence.
- 1972
  - *Practice:* Yuen runs through the various popular orderings of Walsh functions, and the most likely users of each.
  - *Algorithm:* Manz presents a fast Walsh transform (FWT) structure that takes data in bit-reversed order and produces the result in sequency order.
- 1973
  - *Algorithm:* Carl and Swartwood present an algorithm for taking in-order data to bit-reversed sequency results. Interestingly, each stage performs exactly the same operations on exactly the same element positions.
  - *Application:* Edwards discusses the synthesis of logic-gate design by transforming the desired function into a Walsh series and operating on that.
  - *Application:* Corrington solves differential and integral equations by converting the equations to a truncated Walsh series then working on the Walsh representation.
- 1976
  - *Algorithm:* Larsen presents a FWT complementary to Manz, taking data in normal order and returns the result in bit-reversed sequency order.
- 1977
  - *Application:* Yuen proposes using Walsh transforms to test random number sequences.
  - *Application:* Cohn and Lempel show a relationship between LFSR sequences and Walsh-Hadamard matrices. This can be exploited and a fast Walsh algorithm applied to identify the signal "phase" from a given LFSR.
  - *Algorithm:* Brown presents a fast Walsh transform (FWT) which takes in-order data to in-order sequency results.
- 1978
  - *Application:* Shih and Han show how Walsh functions can be used to help solve first-order partial differential equations.
  - *Review:* Beer has some comments on Beauchamp's book.
- 1981
  - *Tutorial:* Beer provides an introduction to the theory and application of Walsh transforms.
- 1984
  - *Text:* Beauchamp, the book. This has a lot good information -- especially the implementation diagrams in sect. 2.3 -- but one must stiff be wary of careless errors present in the earlier version(see Beer).
- 1987
  - *Theory:* Tezuka presents a Walsh spectral test for analyzing GFSR sequences.
  - *Application:* Feldman also tests random number sequences.
- 1988
  - *Application:* Guo-Zhen and Massey relate Walsh transforms to correlation in combining functions.

1969 -- Shanks

Shanks, J. 1969. Computation of the Fast Walsh-Fourier Transform. *IEEE Transactions on Computers*. C-18: 457-459.

"*Abstract* -- The discrete, orthogonal Walsh functions can be generated by a multiplicative iteration equation. Using this iteration equation, an efficient Walsh transform computation algorithm is derived which is analogous to the Cooley-Tukey algorithm for the complex-exponential Fourier transform."

1970 -- Henderson

Henderson, K. 1970. Comment on "Computation of the Fast Walsh-Fourier Transform." *IEEE Transactions on Computers*. C-19: 850-851.

"Abstract -- The matrix form of the Walsh functions . . . can be generated by the modulo-2 product of two generating matrices: the natural binary code, and the transpose of the bit-reversed form of the first. As a result, the coefficients of the Walsh transform occur in bit-reversed order. By simply reordering the Walsh functions themselves to correspond to generation by the product of two such code matrices, neither or both in bit-reversed form, the Walsh coefficients occur in natural order."

1970 -- Yuen

Yuen, C. 1970. Walsh Functions and Gray Code. *Proceedings of the Walsh Function Symposium*. 68-73.

"Gray code is a natural way of ordering binary vectors in dyadic space, hence it appears frequently in connection with Walsh functions. In Paley's definition of Walsh functions their sequencies are arranged in Gray code."

"While neither Paley or Fine mentioned this, we now know that"

$$\mathrm{pal}(g(i),x) = \mathrm{wal}(i,x)$$

[That is, the ith Walsh sequence is the same as the $g(i)^{th}$ Paley sequence, for Gray code function g(), e.g.: {0,1,3,2,6,7,5,4}.]

1971 -- Kak

Kak, S. 1971. Classification of Random Binary Sequences Using Walsh-Fourier Analysis. *Proceedings of Applications of Walsh Functions*. 74-77. Washington, D.C., 1971.

"This paper presents a straightforward procedure using Walsh functions to determine the pattern in a binary sequence."

". . . classification of data amounts to the computation of structure with respect to some criterion."

"A sequence shall be said to have no pattern or be random if the number of independent amplitudes in the Wash-Fourier transform is equal to the length of the sequence itself, i.e., 2^k."

"The measure of randomness r(s) shall be defined by"

$$r(s) = \text{no. of independent amplitudes of } W(s) / \text{length of the sequence} = i(s) / L(s)$$

"The number of independent amplitudes of W(s) shall equal the number of its component Walsh waves." *[The number of non-zero terms?]*

1972 -- Yuen

Yuen, C. 1972. Remarks on the Ordering of Walsh Functions. *IEEE Transactions on Computers*. C-21: 1452.

". . . Walsh functions are characters of the dyadic group, which is the group of binary vectors under bitwise addition modulo 2." "Since there is no natural ordering of the dyadic group, there is no natural ordering for Walsh functions."

"At least three different orderings of Walsh functions are known to have been used. The ordering originally employed by Walsh [6] is commonly known as 'sequency ordering.' This is characterized by the fact that the *i*th function wal(*i*,x) has *i* sign changes in the interval x in [0,1). As the number of sign changes is used as a generalized frequency [7], the ordering is favored by communications engineers . . . ."

"A different ordering used by Paley [10] is characterized by the fact that in this form Walsh functions can be readily expressed as products of Rademacher functions. Most mathematical discussions use this form [11]."

"A third ordering, proposed by Henderson . . . is simply Paley's ordering in reversed binary. It is the ordering that emerges if one computes fast Walsh transforms without sorting, hence it is computationally advantageous."

"The conversion from Paley's ordering to Walsh's ordering is the same as conversion from Gray code to binary . . . ." "Just as we form a path of minimum length on the real line if we order integers by their arithmetic value, Gray code orders points in dyadic space into a path of minimum length."

1972 -- Manz

Manz, J. 1972. A Sequency-Ordered Fast Walsh Transform. *IEEE Transactions on Audio and Electroacoustics*. AU-20: 204-205.

"The only drawback with the fast Hadamard transform (FHT) is that those matrices that possess a simple recursive formula and, therefore, a fast algorithm, are not capable of directly producing the output coefficients ordered by increasing frequency [4], [5]. Sequency, as define by Harmuth [6, p. 50], is one-half the average number of zero crossings per unit time interval. The ordering of the output coefficients of a typical FHT is called dyadic or Paley ordering [5]."

"In order to convert from dyadic to sequency ordering, the output coefficient ordering must be decoded by using a Gray code-to-binary decoder [5], [7]."

"By suitably modifying the FHT approach, a sequency-ordered FWT can be computed that shares all of the good properties of the FHT but eliminates the Gray code decoding."

"In the modified FHT, the input *must* be bit reversed prior to the actual transformation."

1973 -- Carl and Swartwood

Carl, J. and R. Swartwood. 1973. A Hybrid Walsh Transform Computer. *IEEE Transactions on Computers*. C-22(7): 669-672.

"Good [5] developed a matrix factorization technique that leads to a fast transform algorithm."

"The . . . factorization results in a flow diagram for a computation algorithm that has the form of Fig. 1. Note the recursive structure of the algorithm: the interconnections of successive layers are identical."

"These results are easily extended to higher dimensions, but it is simpler to note that higher order transforms can be expressed as on-dimensional transforms with the inputs and outputs relabeled.

1973 -- Edwards

Edwards, C. 1973. The Application of the Rademacher/Walsh Transform to Digital Circuit Synthesis. *Theory and Applications of Walsh Functions*. The Hatfield Polytechnic; June 28th and 29th, 1973.

". . . certain operations in the Rademacher/Walsh transform domain may be used to facilitate logic synthesis. These operations are easily carried out with the aid of a small digital processor."

"If optimum synthesses of predominantly first-order functions are available then logic synthesis may be carried out wholly in the transform domain."

"The application of the Rademacher/Walsh transform to conventional logic synthesis serves to emphasize the, often neglected, role that exclusive-OR function plays in the completion of Boolean functions. Indeed, although the use of the exclusive-OR gate is usually avoided in conventional logic methods, it appears that their use is essential to the generation of elegant syntheses."

1973 -- Corrington

Corrington, M. 1973. Solution of Differential and Integral Equations with Walsh Functions. *IEEE Transactions on Circuit Theory*. CT-20(5): 470-476.

"*Abstract* -- Any well-behaved periodic waveform can be expressed as a series of Walsh functions. If the series is truncated at the end of any group of terms of a given order, the partial sum will be a staircase approximation to the waveform. The height of each step will be the average value of the waveform over the same interval.

"If a zero-memory nonlinear transformation is applied to a Walsh series, the output series can be derived by simple algebraic processes. The coefficients of the input series will change, but there will be no new terms not in the original groups.

"Nonlinear differential and integral equations can be solved as a Walsh series, since the series for derivatives can always be integrated by simple table lookup. The differential equation is solved for the highest derivative first and the result is then integrated the required number of times to give the solution."

1976 -- Larsen

Larsen, H. 1976. An Algorithm to Compute the Sequency Ordered Walsh Transform. *IEEE Transactions on Acoustics, Speech, and Signal Processing*. ASSP-24: 335-336.

"The coefficients from Shanks' algorithm are in 'dyadic,' or Paley order, which is merely the naturally ordered coefficients after bit reversal. In many applications the most convenient ordering is with the coefficients corresponding to the Walsh functions arranged by increasing number of zero crossings. This is known as sequency ordering. Sequency ordering can be obtained from dyadic ordering by a permutation based on the gray code [3], [4]."

"In 1972, however, Manz [5] introduced a sequency ordered FWT which eliminates gray code reordering by suitably modifying the basic FHT structure." "Manz's algorithm requires the input data in bit-reversed order, returning the coefficients in sequency order."

"The algorithm presented here is complementary to the one developed by Manz, and has all the same advantages, namely that it is in place, and is its own inverse. It differs from Manz's algorithm in that it has a decimation-in time structure, and accepts data in normal order, returning the coefficients in bit-reversed sequency order."

1977 -- Yuen

Yuen, C. 1977. Testing Random Number Generators by Walsh Transform. *IEEE Transactions on Computers*. C-26(4): 329-333.

"*Abstract* -- A truly random sequence of numbers has an asymptotically flat Walsh power spectrum. This fact is used to devise a new test for the randomness of the output of random number generators."

"One essential randomness test is that of uncorrelatedness, i.e., that the autocorrelation of the sequence is approximately a *d*-function."

"A property equivalent to uncorrelatedness is that the power spectrum be flat."

"In this paper we propose another randomness test equivalent to the correlation test: that the Walsh power spectrum be flat." "Thus, testing the flatness of the spectrum is equivalent to testing for uncorrelatedness of the values of x."

". . . the band spectrum estimate can also be evaluated by spectrum averaging . . . ."

"With segment averaging there is no longer any difficulty with core requirements. When we wish to test a sequence of 2^n values, we would read in, or generate 2^{n-m} values at a time, compute the 2^{n-m}-point fast Walsh transform of the segment, square, and add the squares to the 2^{n-m} memory locations which have been initially set to zero. After all 2^m segments have been processed these memory locations will contain the band spectrum estimate, and we can then proceed to examine if it is consistent with a flat S."

"Another possible additional test is that we permute the random numbers in some way before Walsh transformation. Given a truly random sequence, we should still get a flat spectrum regardless of what permutation was tried."

1977 -- Cohn and Lempel

Cohn, M. and A. Lempel. 1977. On Fast M-Sequence Transforms. *IEEE Transactions on Information Theory*. IT-23: 135-137.

"An *M*-sequence is a binary sequence generated by a linear feedback shift-register whose characteristic polynomial is primitive. An *M*-sequence can be shown to have an impulse-like autocorrelation function [1]; for this reason *M*-sequences are often called 'pseudo-noise sequences,' and their distinct cyclic permutations, or phases, form a useful signalling alphabet [1]-[3]. A drawback is that correlation-detection at the receiver end [is very costly]." "The cost of this computation can be drastically reduced by exploiting the equivalence between the *M*-sequence matrix and the Walsh-Hadamard matrix." "[This] has been successfully used in the rapid decoding of first-order Reed-Muller codes [3]-[5]."

1977 -- Brown

Brown, R. 1977. A Recursive Algorithm for Sequency-Ordered Fast Walsh Transforms. *IEEE Transactions on Computers*. C-26(8): 819-822.

"The FWT [Fast Walsh Transform] may be developed by analog with the Cooley-Tukey algorithm [7] for the fast Fourier transform (FFT). Implementation of the FWT results in a reduction of the number of computations from N^2 to $N\log_2 N$ when applied to a sampled data set of *N* elements. Shanks [8] described such an FWT, which used the multiplicative iteration equations for calculating the discrete Walsh functions. However, this algorithm produced FWT in dyadic or natural order instead of the sequency order more useful for spectral analysis. (Sequency, first defined by Harmuth [2], is one-half the average number of zero-crossings per unit interval.) A similar FWT algorithm, based on the Cooley-Tukey algorithm, was developed by Manz [9] which produced sequency-ordered transforms but required bit-reversal of the input data."

1978 -- Shih and Han

Shih, Y. and J. Han. 1978. Double Walsh series solution of first-order partial differential equations. *International Journal of Systems Science*. 9(5): 569-578.

"A double Walsh series is introduced to represent approximately functions of two independent variables, and is then applied to analyze single as well as simultaneous first-order partial differential equations. The solutions for the coefficient matrices can be obtained directly from Kronecker product formulae, which are suitable for computer computation. An example for a single first-order partial differential equation is solved by a double Walsh series approximation with satisfactory results."

1978 -- Beer

Beer, T. 1978. (Book Review.) *Search*. 9: 421.

". . . the Walsh transform proceeds by additions and subtractions and is thus far more efficient in use of computer time than the Fourier transform. There has long been a need for a readable, comprehensive text on Walsh functions and their applications. This need continues unabated."

"It is inexcusable that Beauchamp, who is described as Director of Computer Services at the University of Lancaster, presents a volume full of incorrect and inefficient computer programs."

1981 -- Beer

Beer, T. 1981. Walsh transforms. *American Journal of Physics*. 49(5): 466-472.

"Walsh functions are an orthogonal set of square-wave functions that arise when dealing with digital data. The Walsh transform and inverse Walsh transform are easy to calculate by hand, and can be very quickly done on digital computers. Examples of the uses of Walsh transform include . . . the rapid solution of nonlinear differential equations."

". . . I have found that undergraduate students experience a great amount of difficulty in understanding the concept of digital Fourier transforms. The study of Walsh transforms provides an excellent introduction to this, because their simplicity enables calculations to be made by hand . . . ."

1984 -- Beauchamp

Beauchamp, K. 1984. *Applications of Walsh and Related Functions*. Academic Press.

This is currently the one book with the most information in one place (but see Beer for a review of the earlier version).

Perhaps the most useful section is 2.3, which gives a number of Fast Walsh Transforms (FWT's) in graphic "butterfly" diagrams.

A *butterfly* is the fundamental operation in many fast transforms, and shows the processing of two elements in one stage of the transform. For FWT's, typically the sum and difference of the two elements are calculated. Frequently, these operations conveniently occur *in place*, that is, the results are placed back into *the same* storage elements.

Various butterfly diagrams are given, including:

- Fig. 2.3, "A flow diagram for a sequency-ordered Walsh transform," but apparently produces the dyadic (Paley) ordering.
- Fig. 2.6, "A dyadic-ordered Walsh transform," shows input in sequential order, but apparently should actually be in bit-reversed order.
- Fig. 2.7 is "A natural-ordered Walsh transform."
- Fig. 2.8 is "A Manz sequency-ordered Walsh transform."
- Fig. 2.9 is "A Larsen sequence-ordered Walsh transform."

along with a number of others.

1987 -- Tezuka

Tezuka, S. 1987. Walsh-Spectral Test for GFSR Pseudorandom Numbers. *Communications of the ACM*. 30(8): 731-735.

"*ABSTRACT: By applying Weyl's criterion for k-distributivity to GFSR sequences, we derive a new theoretical test for investigating the statistical property of GFSR sequences.*"

"It is well known that Walsh-spectral analysis provides certain information about the dyadic correlation of a sequence [1]. This paper shows that Walsh-spectral analysis can also be exploited to investigate the *k*-distribution of the sequence."

1987 -- Feldman

Feldman, F. 1987. Fast Spectral Tests for Measuring Nonrandomness and the DES. *Advances in Cryptology -- CRYPTO '87*. 243-254. Springer-Verlag.

"*Abstract* -- Two spectral tests for detecting nonrandomness were proposed in 1977. One test, developed by J. Gait [1], considered properties of power spectra obtained from the discrete Fourier transform of finite binary strings."

"Another test, developed by C. Yuen [2], considered analogous properties for the Walsh transform. In estimating variance of spectral bands, Yuen assumes the spectral components to be independent. Except for the special case of Gaussian random numbers, this assumption introduces a significant error into his estimate."

"A new test, based on an evaluation of the Walsh spectrum, is presented here. This test extends the earlier test of C. Yuen."

"We prove that our measure of the Walsh spectrum is equivalent to a measure of the skirts of the logical autocorrelation function. It is clear that an analogous relationship exists between Fourier periodograms and the circular autocorrelation function."

1988 -- Guo-Zen and Massey

Guo-Zhen, X. and J. Massey. 1988. A Spectral Characterization of Correlation-Immune Combining Functions. *IEEE Transactions on Information Theory*. 34(3): 569-571.

"*Abstract* -- It is shown that a Boolean combining function *f(x)* of *n* variables is *m*th-order correlation immune if and only if its Walsh transform *F(w)* vanishes for all *w* with Hamming weights between 1 and *m*, inclusive."

Terry Ritter, his current address, and his top page.

Last updated: 1996-08-15

Linear Complexity: A Literature Survey

Research Comments from Ciphers By Ritter

Terry Ritter

The *linear complexity* (LC) of a sequence is the size in bits of the shortest linear feedback shift register (LFSR) which can produce that sequence. The measure therefore speaks to the difficulty of generating -- and perhaps analyzing -- a particular sequence.

Randomness can be seen as the size of the smallest program to produce a given sequence. But linear complexity is the size of a LFSR "processor" to produce a sequence, and there is an algorithm (Berlekamp-Massey) to measure the LC. So the resulting LC value might be used to measure one view of randomness.

Contents

- 1969
  - *Application*: Massey presents the article linking the Berlekamp algorithm to LFSR reconstruction.
- 1971
  - *Theory*: Groth describes nonlinear SR generators with high complexity.
- 1976
  - *Theory*: Key gives a method for predicting the complexity of a sequence produced by SR's with nonlinear operations. It also speaks highly of Geffe's generator, which is subsequently broken completely.
  - *Tutorial*: Gustavson analyzes Berlekamp-Massey for computation costs.
- 1979
  - *Theory*: Welch and Scholtz an algorithm for developing rational approximations using continued fractions ends up virtually equivalent to Massey-Berlekamp.
- 1981
  - *Text*: Clark and Cain.
- 1982
  - *Theory*: Ferguson and Forcade construct "multidimensional" generalizations of the Euclidean algorithm.
- 1983
  - *Text*: Blahut.
- 1984
  - *Theory*: Mandelbaum develops an arithmetic analog to Berlekamp-Massey.
  - *Theory*: Cheng, like Welch and Sholtz before, also reports a deep similarity between continued fractions and Berlekamp-Massey.
- 1985
  - *Theory*: Brynielsson how to calculate the linear complexity of a function in GF(2*) (instead of GF(2)) to avoid trading off linear complexity for correlation immunity.
  - *Tutorial*: Herlestam addresses linear complexity.
- 1987
  - *Theory*: Siegenthaler and Forre shows how to construct sequences of high complexity.
  - *Theory*: Dornstetter shows that Berlekamp-Massey can be derived from a version of Euclid's algorithm.
  - *Application*: Rueppel and Staffelbach show how to compute the linear complexity of a resulting sequence when the LC of each constituent sequence is known.
  - *Theory*: Imamura and Yoshida present an alternate and perhaps easier derivation of Berlekamp-Massey.
- 1989
  - *Theory*: Chan and Games: Linear complexity is generally a measure of linear *span*, related to *quadratic* span, which can be unexpectedly very much smaller than the linear value.
- 1994
  - *Algorithm*: Fuster-Sabater and Caballero-Gil compute a lower bound to linear complexity for a given design, using binary string operations.
  - *Theory*: Massey and Serconek address the opportunity to measure the linear complexity of nonlinearly-filtered LFSR sequences using the discrete Fourier transform (DFT).
  - *Algorithm*: Fitzpatrick finds an algorithm similar to Berlekamp-Massey which appears more suitable for a parallel hardware implementation.
- 1995
  - *Algorithm*: Fleischmann presents a modified Berlekamp-Massey which extends the model sequence in both directions around any given data bit.

1969 -- Massey

Massey, J. 1969. Shift-Register Synthesis and BCH Decoding. *IEEE Transactions on Information Theory*. IT-15(1): 122-127.

"*Abstract* -- It is shown in this paper that the iterative algorithm introduced by Berlekamp for decoding BCH codes actually provides a general solution to the problem of synthesizing the shortest linear feedback shift register capable of generating the prescribed finite sequence of digits." "The equivalence of the decoding problem for BCH codes to a shift-register synthesis problem is demonstrated . . . ."

1971 -- Groth

Groth, E. 1971. Generation of Binary Sequences With Controllable Complexity. *IEEE Transactions on Information Theory*. IT-17: 288-296.

"*Abstract*--Complexity of a binary sequence is measured by the amount of the sequence required to define the remainder. It is shown that, while maximum length ($L = 2^r - 1$) sequences from *r*-stage linear feedback generators have minimum complexity, it is a simple matter to use such sequences as bases for deriving other more complex sequences of the same length."

"As a preliminary, it is important to recognize that any sequence can be generated by one or more linear sequences." ". . . it is apparent that the adjectives, linear or nonlinear, applied frequently to sequences have no real meaning. Generators may be characterized as linear or nonlinear, but sequences may not."

1976 -- Key

Key, E. 1976. An Analysis of the Structure and Complexity of Nonlinear Binary Sequence Generators. *IEEE Transactions on Information Theory*. IT-22: 732-736.

"**Abstract**--A method of analysis is presented for the class of binary sequence generators employing linear feedback shift registers with nonlinear feed-forward operations. This class is of special interest because the generators are capable of producing very long 'unpredictable' sequences."

"This paper presents a method of analysis based on Galois field theory that enables one to predict the generator complexity resulting from nonlinear operations. Moreover the theory provides a conceptually simple basis for synthesizing devices with the desired characteristics, and is readily extendable to a larger class of more complex generators."

1976 -- Gustavson

Gustavson, F. 1976. Analysis of the Berlekamp-Massey Linear Feedback Shift-Register Synthesis Algorithm. *IBM Journal of Research and Development*. 20: 204-212.

"**Abstract**: An analysis of the Berlekamp-Massey Linear Feedback Shift-Register (LFSR) Synthesis Algorithm . . . ." ". . . we present Massey's algorithm and indicate which steps contribute to the computation cost. We then prove what the minimum, average, and maximum computation costs are in terms of the numbers of multiplications and additions."

1979 -- Welch and Scholtz

Welch, L. and R. Sholtz. 1979. Continued Fractions and Berlekamp's Algorithm. *IEEE Transactions on Information Theory*. IT-25(1): 19-27.

"*Abstract* -- Theorems are presented concerning the optimality of rational approximations using non-Archimedean norms. The algorithm for developing the rational approximations is based on continued fraction techniques and is virtually equivalent to an algorithm employed by Berlekamp for decoding BCH codes. Several variations of the continued fraction technique and Berlekamp's algorithm are illustrated on a common example.

1981 -- Clark and Cain

Clark, G. and J. Cain. 1981. *Error-Correction Coding for Digital Communications*. Plenum Press.

Includes Berlekamp-Massey, and a lot more.

1982 -- Ferguson and Forcade

Ferguson, H and R. Forcade. 1982. Multidimensional Euclidean algorithms. *Journal Fur Dir Reine Und Angewandte Mathematik*. Walter de Gruyter & Co.

"**Introduction.** Given a pair (*x*, *y*) of positive real numbers, then one iteration of the Euclidean algorithm replaces the larger number by its least non-negative residue modulo the smaller number. If the two numbers are linearly dependent, the repetition of this process will eventually terminate with a pair in which one of the elements is zero. (If the original pair were integers, then the remaining non-zero element is their greatest common divisor.)"

"Another property of the Euclidean algorithm, fundamental to the study of continued fractions, is that it produces increasingly good rational approximations to the original pair of real numbers."

"We will construct an iterative algorithm for *n*-tuples, generalizing both the terminating and approximating features of the Euclidean algorithm. Thus, if the original *n*-tuple of elements are Z-linearly dependent, the algorithm will necessarily terminate and discover the Z-relation among the elements of the original *n*-tuple. If the original *n*-tuple elements are not Z-linearly dependent, then the algorithm will "absolutely approximate" by producing lattice points arbitrarily close to the line generated by the original *n*-tuple."

"We emphasize that a major difficulty in the problem of constructing a generalization of the Euclidean algorithm is to give an *iterative* algorithm."

1983 -- Blahut

Blahut, R. 1983. *Theory and Practice of Error Control Coding*. Addison-Wesley.

Covers decoding BCH codes both with Berlekamp-Massey and Euclidean algorithms, plus much more.

1984 -- Mandelbaum

Mandelbaum, D. 1984. An Approach to an Arithmetic Analog of Berlekamp's Algorithm. *IEEE Transactions on Information Theory*. IT-30(5): 758-762.

"In 1968 Berlekamp [1] introduced an iterative procedure to determine a polynomial fraction *A(x) / B(x)* in which the coefficients are members of a field that, when divided, yields a given polynomial sequence *S(x)* in which the coefficients are, of course, members of the same field." "Massey gave another elegant proof of this procedure in terms of shift-registers [2]. Similar results were then obtained with the Euclidean algorithm and continued fractions [3]-[6]. Since continued fractions can be utilized with both polynomials and numbers from the real field, using an algorithm similar to the Berlekamp algorithm with real binary numbers was investigated."

"The class of arithmetic codes to which the proposed algorithm will apply is usually termed arithmetic residue codes and is a natural coding method for residue computers [12], which multiply and add quickly by using the Chinese remainder theorem." "These residue codes are the arithmetic analogs of the Reed-Solomon codes, which also can be decoded by the Euclidean algorithm or Berlekamp's algorithm."

1984 -- Cheng

Cheng, U. 1984. On the Continued Fraction and Berlekamp's Algorithm. *IEEE Transactions on Information Theory*. IT-30(3): 541-544.

"*Abstract* -- Continued fraction techniques are equivalent to Berlekamp's algorithm."

"Implementation of Berlekamp's algorithm by continued fraction technique was introduced by Reed *et. al.* [2]. In [3], Welch and Scholtz studied the equivalence between continued fraction and Berlekamp's algorithm. We will show that all entities in Berlekamp's algorithm [1] can be related to those in the continued fraction."

1985 -- Brynielsson

Brynielsson, L. 1985. On The Linear Complexity of Combined Shift Register Sequences. *Advances in Cryptology -- EUROCRYPT* '85. 156-160. Springer-Verlag.

"Many proposed keystream generators consist of a number of binary maximum length shift registers combined by a nonlinear binary function. The registers guarantee a long period and the nonlinear function destroys the linearity i.e. it gives the output sequence a large linear complexity [1] (*linear* equivalent [2]). In order to avoid correlation attacks the function should also be correlation immune [3] i.e. the output sequence should be statistically independent of the various inputs. There is however a trade off between the linear complexity and the order of correlation immunity . . . . The reason for this is that in the binary field GF(2) there are too few functions."

"In the field GF(2*) the situation is different. For instance, the polynomial function $x+y+3xy+2(x^2y+xy^2)+x^2y^2$ in GF(4) is both nonlinear and correlation immune. In order to validate such a function one must be able to calculate its linear complexity. That is the purpose of this paper."

1985 -- Herlestam

Herlestam, T. 1985. On Functions of Linear Shift Register Sequences. *Advances in Cryptology -- EUROCRYPT* '85. 119-129. Springer-Verlag.

"**Abstract**. This paper is intended as an overview, presenting several results on the linear complexity of sequences obtained from functions applied to linear shift register sequences."

"Two well-known models for shift registers are in use. The Fibonacci model consists of cascaded memory boxes. The contents of each box is multiplied by a feedback coefficient before being taken to a common summing device to produce the feedback element."

"In the Galois model, adders are inserted between the memory boxes, the system output is multiplied by the feedback coefficients . . . and the products are taken to the adders."

"In both cases, the same shift register recurrence is obtained."

"Three different methods for handling this recurrence are in use. The linear algebraic (matrix) method is the most commonly used (e.g., Golomb (1967)) . . . ."

"Rewriting the shift register recurrence . . . we can apply the classical technique as use by Selmer (1966) and Key (1976) among others."

"Finally, the generating function method, used by Zierler (1959), can be applied to the shift register recurrence.

1987 -- Siegenthaler and Forre

Siegenthaler, T. and R. Forre. 1987. Generation of Binary Sequences with Controllable Complexity and Ideal *r*-Tuple Distribution. *Advances in Cryptology -- EUROCRYPT* '87. 15-23.

"**Abstract**. A key stream generator is analyzed which consists of a single linear feedback shift register (LFSR) with a primitive connection polynomial and a nonlinear feedforward logic." ". . . a simple condition imposed on these logics ensures an ideal *r*-tuple distribution for these keystreams."

"Groth [2] proposed a layered structure for the feedforward logic to control the linear complexity of the generated keystream. This arrangement generates keystreams of large linear complexities, however, the statistics of these keystreams is hard to control. Rueppel suggested [3] *f the dissertation* j a simply realisable and therefore practically useful class of feedforward logics such that a lower bound for the keystream's linear complexity is guaranteed."

"Our analysis is based on Brynielsson's powerful Theorem 1 from which the linear complexity for every polynomial *f* applied to a maximum length sequence can be computed even if we use it only for a function . . . ."

1987 -- Dornstetter

Dornstetter, J. 1987. On the Equivalence Between Berlekamp's and Euclid's Algorithms. *IEEE Transactions on Information Theory*. IT-33(3): 428-431.

"*Abstract* -- It is shown that Berlekamp's iterative algorithm can be derived from a normalized version of Euclid's extended algorithm."

"The similarity between Berlekamp's iterative algorithm [1] and the extended Euclidean algorithm [7] has been previously noticed by several authors [3], [4]." "The original version of the iterative algorithm has been simplified by Massey [2]. We shall show in Section II that all partial results generated by this simplified version are in agreement (to a reciprocation and a normalization factor) with those given by Euclid's algorithm."

1987 -- Rueppel and Staffelbach

Rueppel, R. and O. Staffelbach. 1987. Products of Linear Recurring Sequences with Maximum Complexity. *IEEE Transactions on Information Theory*. IT-33(1): 124-131.

"*Abstract* -- Conditions are derived which guarantee that products of linear recurring sequences attain maximum linear complexity. It is shown that the product of any number of maximum-LENGTH GF(q) sequences has maximum linear complexity, provided only the degrees of the corresponding minimal polynomials are distinct and greater than two."

"A common type of running-key generator employed in stream cipher systems consists of *n* (mostly maximum-length) linear feedback shift registers (LFSR's) whose output sequences are combined in a nonlinear function *F* to produce the key stream." "One of the major objectives of *F* is to increase the linear complexity of the key stream such that the synthesis of a linear equivalent of the running-key generator (e.g., by using the Berlekamp-Massey LFSR synthesis algorithm [10]) becomes computationally feasible."

"We will stipulate that whenever we write of a product of two or more sequences we mean the termwise product of those sequences."

1987 -- Imamura and Yoshida

Imamura, K. and W. Yoshida. 1987. A Simple Derivation of the Berlekamp-Massey Algorithm and Some Applications. *IEEE Transactions on Information Theory*. IT-33(1): 146-150.

"The algorithm discovered by Berlekamp [1] for decoding BCH codes is very elegant. Massey [2] showed that Berlekamp's algorithm is best interpreted as an efficient recursive method for finding the shortest linear feedback shift register (LFSR) that generates a given sequence. Since Massey's interpretation is very useful, the algorithm is often called the Berlekamp-Massey algorithm."

"The derivation of the Berlekamp-Massey algorithm, however, seems to be rather difficult, and why it works is not so easy to understand [3], [4]." "The main purpose of this correspondence is to present a new method for deriving the Berlekamp-Massey algorithm . . . to make this important algorithm more easily understandable." "1) Find a general rule how the length of the LFSR grows with the sequence length and find a necessary and sufficient condition on the LFSR to be unique." "2) Find a recursive algorithm for updating the LFSR."

1989 -- Chan and Games

Chan, A. and R. Games. 1989. On the Quadratic Spans of Periodic Sequences. *Advances in Cryptology -- CRYPTO* '89. 82-89. Springer-Verlag.

"The length of a shortest FSR [feedback shift register] that generates the sequence is called the *span* of the sequence.

"Because of its tractability, most attention has been focused on determining the *linear span* of a sequence -- the length of the shortest linear FSR that generates the sequence."

"A sequence with very large linear span may be generated by a much shorter FSR if nonlinear terms are allowed in the feedback function."

1994 -- Fuster-Sabater and Caballero-Gill

Fuster-Sabater, A. and P. Caballero-Gil. 1994. On the Linear Complexity of Nonlinearly Filtered PN-sequences. *Advances in Cryptology -- ASIACRYPT* '94. 80-90. Springer-Verlag.

"**Abstract**. A method of analysis for the linear complexity of nonlinearly filtered PN-sequences is presented. The procedure provides a general lower bound for the linear complexity and an algorithm to improve it. The results obtained are valid for any nonlinear function with a unique term of maximum order and for any maximal-length LFSR. This work, which has as its starting point 'the root presence test' by Rueppel, is based on the handling of binary strings instead of determinants in a finite field."

"Groth [1] presented the linear complexity as a controllable parameter with the order of F. Nevertheless, in his work there is no explicit mention to the degeneracies which may occur in the linear complexity of the produced sequence."

"Key [2] established the relationship between the minimal polynomial roots required to represent the keystream generator and the linear complexity of the generated sequence. This result let Rueppel [7] state the so called '*root presence test*' for the product of distinct phases of a PN-sequence."

". . . this paper proposes an algorithm to compute a lower bound for the complexity by using exclusively logic operations (OR, AND)."

1994 -- Massey and Serconek

Massey, J. and S. Serconek. 1994. A Fourier Transform Approach to the Linear Complexity of Nonlinearly Filtered Sequences. *Advances in Cryptology -- CRYPTO* '94. 332-340. Springer-Verlag.

"By exploiting 'Blahut's theorem,' which states that the linear complexity of an *N*-periodic sequence in $GF(q)^N$ and the Hamming weight of its *frequency-domain* associate are equal, we use Discrete Fourier Transform (DFT) techniques here to study the linear complexity of nonlinear filterings of PN(pseudo-noise)-sequences."

1994 -- Fitzpatrick

Fitzpatrick, P. 1994. New time domain errors and erasures decoding algorithm for BCH codes. *Electronics Letters*. 30(2): 110-111.

"A new algorithm is presented for solving the key equation that simultaneously computes the error locator polynomial and the errata evaluator polynomial. The algorithm is similar to the Berlekamp algorithm, but is more symmetrical in its treatment of the iterated pairs of polynomials making it particularly well suited to a highly parallel hardware implementation."

1995 -- Fleischmann

Fleischmann, M. 1995. Modifiied Berlekamp-Massey algorithm for two-sided shift-register synthesis. *Electronics Letters*. 31(8): 605-606.

"*Introduction:* Massey introduced in [3] an algorithm for calculating the shortest LFSR which generates a given finite sequence . . . . In [3] the synthesizing process is one-sided and a definite beginning or end of the sequence is needed. However, some applications may be imagined where prior knowledge of neither a sequence border nor the size of the sequence are available. As a result, the algorithm has to start somewhere within the sequence, and has to develop the LFSR using a two-sided synthesising process."

Terry Ritter, his *current address*, and his *top page*.

Last updated: 1996-08-15

RNG Surveys: A Literature Survey

Research Comments from Ciphers By Ritter

Terry Ritter

There are so many different random number generators (RNG's), and so many claims made for them, that many people have tried to expose the mysteries -- including me!

These are various surveys of **pseudo(!)**-random number generator schemes from various different points of view.

Contents

- 1983
  - Ripley gives us an early discussion of RNG designs and testing; no particular RNG is proposed. Included are approaches for producing random deviates in various distribution.
- 1989
  - Lewis and Orav present a full chapter on RNG's in their simulation text. They survey various RNG designs and promote graphic testing of the distributions in the resulting designs.
  - L'Ecuyer presents what is apparently another version of L'Ecuyer 1990. This is a survey of the various types of RNG, and no particular implementation is promoted.
- 1990
  - L'Ecuyer discusses generators for simulation use. Very like L'Ecuyer 1989. Again, this is a survey of the various types of RNG, and no particular implementation is promoted.
  - James discusses generators for Monte Carlo use. This is another survey of RNG type, this time with some examples of famous LCG multipliers. Several examples of famous current designs are given in FORTRAN, but not tested here.
  - Lagarias discusses generators with respect to number theory and cryptography. Very theoretical, but touches on one-way functions, unpredictability, and computational complexity issues.
- 1991
  - Zeng, Yang, Wei and Rao discuss bit generators for stream-cipher cryptography. Again, a survey of different types of RNG, but here including approaches intended to make the RNG difficult to analyze from its sequence.
  - Ritter I survey RNG's for use in cryptography, with extensive references.

1983 -- Ripley

Ripley, B. 1983. Computer Generation of Random Variables: A Tutorial. *International Statistical Review*. 51: 301-319.

"**Summary.** Users of small and personal computers do not have access to the libraries of generators of random variables provided by experts which have been common on large computer installations. This tutorial provides the background needed by user-programmers to replace the often inadequate pseudorandom number generators supplied with small computers, and to program simple yet efficient methods to generate samples from exponential, normal, gamma, Poisson, ..., distributions for use in simulation studies."

1. Rationale
2. Pseudorandom numbers
3. Testing pseudorandom-number generators
4. Continuous distributions
   1. Inversion
   2. Transformations
   3. Acceptance-rejection
   4. Ratio methods
   5. Composition
5. Conclusions

1989 -- Lewis and Orav

Lewis, P. and E. Orav. 1989. *Simulation Methodology for Statisticians, Operations Analysts, and Engineers*. Wadsworth and Brooks: Pacific Grove, CA.

In Chapter 4 they introduce the use of "scatter plots" to examine generators.

Chapter 5. Uniform Pseudo-Random Variable Generation.

1. Introduction: Properties of Pseudo-Random Variables
   1. A uniform marginal distribution
   2. Independence of the uniform variate
   3. Repeatability and portability
   4. Computational speed
2. Historical Perspectives
3. Current Algorithms
   1. Congruential Generators
   2. Shift-Register Generators
   3. Fibonacci Generators
   4. Combinations of Generatrtes (Shuffling)
   5. Generators in Packages
4. Recommendations for Generators
5. Computational Considerations
6. The Testing of Pseudo-Random Number Generators
7. Conclusions on Generating and Testing Pseudo-Random Number Generators
   1. "Testing is unnecessary in the sense that very good pseudo-random number generators are available."
   2. "Testing is *necessary* in the sense that bad pseudo-random number generators exist on many computer systems and are commonly used."
   3. "It is preferable to substitute a good pseudo-random number generator with documented properties for a pseudo-random number generator you known nothing about. And even if you follow [this rule], it is wise to use some of the graphical testing methods given in this book to make sure that the implementation is correct."

1989 -- L'Ecuyer

L'Ecuyer, P. 1989. A Tutorial on Uniform Variate Generation. *Proceedings of the 1989 Winter Simulation Conference*. 40-49.

"**ABSTRACT.** In typical stochastic simulations, randomness is produced by generating a sequence of independent uniform variates (usually real-valued between 0 and 1, or integer-valued in some interval) and transforming them in the appropriate way. In this tutorial, we examine practical ways of generating such variates on a computer."

[This paper is very similar to the 1990 version in *Communications of the ACM*, Please refer to that review.]

1990 -- L'Ecuyer

L'Ecuyer, P. 1990. Random Numbers for Simulation. *Communications of the ACM*. 33(1): 85-97.

"On the mind of the average computer user, the problem of generating uniform variates by computer has been solved long ago. After all, every computer system offers one or more function(s) to do so." "These functions *supposedly* return numbers that could be used, for all practical purposes, as if they were the values taken by independent random variables, with a uniform distribution between 0 and 1."

"We focus mainly on efficient and recently proposed techniques for generating *uniform pseudorandom numbers*." Here, 'uniform pseudorandom' means that the numbers behave from the outside as if they were the values of i.i.d. [individually independently distributed] random variables, uniformly distributed over some finite set of symbols. This set of symbols is often a set of integers of the form {0, ..., *m* - 1} and the symbols are usually transformed by some function into values between 0 and 1, to approximate the U(0,1) distribution."

- Views of Randomness
  - Classical Definitions
  - A Framework for PRNGs
  - PT-perfect generators
- Matrix Linear Congruential Recurrences
  - Prime Modulus
  - Composite Modulus
  - Jumping Ahead, Splitting, and Vectorization
  - Implementations
  - Multiple Recursive Generators
  - Lattice Structure and Spectral Test
  - Tausworthe, GFSR, Lagged-Fibonacci
  - Other Variants
- Non-Linear Generators
  - A Class of Generators by Inversion
- Combined Generators
- Transforming into U(0,1) Variates
- Statistical Testing
- Discrepancy

1990 -- James

James, F. 1990. A review of pseudorandom number generators. *Computer Physics Communications*. 60: 329-344. North-Holland.

"This is a brief review of the current situation concerning practical pseudorandom number generation for Monte Carlo calculations. The conclusion is that pseudorandom number generators with the required properties are now available, but the generators actually used are often not good enough. Portable Fortran code is given for three different pseudorandom number generators, all of which have much better properties than any of the traditional generators commonly supplied in most program libraries."

1. General considerations
   1. The motivation and scope of this paper
   2. The three types of generators
      1. Truly random numbers . . . must be produced by a random physical process."
      2. "Pseudorandom numbers . . . are supposed to appear random to someone who doesn't know the algorithm."
      3. "Quasirandom numbers . . . are not designed to appear to be random, but rather to be distributed as uniformly as possible, in order to reduce the errors in Monte Carlo integration."
   3. Desirable properties of a random number generator
      1. Good distribution.
      2. Long period.
      3. Repeatability.
      4. Long disjoint subsequences.
      5. Portability.
      6. Efficiency.
   4. Manufacturer-supplied generators
2. Pseudorandom numbers
   1. Testing good distributions
   2. Pseudorandom generation methods
      1. MLCG [multiplicative linear congruential generator]
      2. Fibonacci
      3. Shift register or tausworthe
   3. Improving simple generators
3. Acceptable pseudorandom generators
   1. The McGill generator super-duper [Marsaglia]
   2. RANECU: the algorithm of TEcuyer
   3. RANMAR: the algorithm of Marsaglia, Zaman and Tsang
   4. ACARRY: the algorithm of Marsaglia and Zaman
4. Conclusions

1990 -- Lagarias

Lagarias, J. 1990. Pseudorandom Number Generators in Cryptography and Number Theory. *Proceedings of Symposia in Advanced Mathematics*. 42: 115-143.

"**Abstract.** This paper describes the close relations that exist between pseudorandom number generators, one-way functions, and private key cryptosystems. It presents a taxonomy of pseudorandom number generators based on number-theoretic constructions and summarizes results on the cryptanalysis of such generators."

1. Introduction
2. A Taxonomy of Number-theoretic Generators
   1. Multiplicative Congruential Generator
   2. 1/P Generator
   3. Power Generator
   4. Discrete Exponential Generator
   5. Kneading Map
   6. Shift-Register Sequences
   7. Cellular Automata
   8. Hashing
   9. Composition
3. What is a Pseudorandom Number Generator?
4. Unpredictability and Pseudorandomness
5. Pseudorandom Bit Generators and One-way Functions
6. Pseudorandom Bit Generators and Private-key Cryptosystems
7. Secure Number-theoretic Pseudorandom Bit Generators
   - RSA bit generator.
   - Modified Rabin bit generator.
   - Discrete exponential bit generator.
8. Cryptanalysis Problems

1991 -- Zeng, Yang, Wei and Rao

Zeng, K., C. Yang, D. Wei and T. Rao. 1991. Pseudorandom Bit Generators in Stream-Cipher Cryptography. *IEEE Computer*. February. 8-17.

"The central problem in stream-cipher cryptography . . . is the difficulty of generating a long unpredictable sequence of binary signals from a short and random key."

"The problem is this: On which basis can one draw the conclusion that the output signals of a certain given keystream generator are hard to predict? No universally applicable and practically checkable criteria have been developed to certify the security of bit generators. For that matter, no general theory of cryptanalysis is known to exist except for an ever-expanding arsenal of concrete attack methods elaborated for various practical purposes."

"Many of the publicly proposed keystream generators. We remind the reader that cracking keystream generators amounts to the same thing as conducting known-plaintext attacks on stream ciphers. Secure communication should resist such attacks."

- Background
  - The one-time pad.
  - Secure pseudorandom bit generators.
- Building Blocks
  - Linear congruence generators (LCGs).
  - Feedback shift registers (FSRs).
  - Nonlinear feedback shift registers.
  - Linear feedback shift registers.
  - The linear complexity issue.
- Attacking Keystream Generators
  - The Siegenthaler correlation attack.
  - The linear consistency attack.
  - The linear syndrome attack.
- Some design techniques
  - Nonlinear feed-forward transformation.
  - Step control.
  - Multiclock systems.

1991 -- Ritter

Ritter, T. 1991. The Efficient Generation of Cryptographic Confusion Sequences. *Cryptologia*. 15(2): 81-139.

"ABSTRACT: A survey of pseudo-random sequence or random number generators (RNG's) for cryptographic applications, with extensive reference to the literature, and seemingly unresolved issues discussed throughout."

1. Introduction
2. Background
3. Basics and Terminology
   1. Randomness
   2. Randomness and Data Compression
   3. Reasoning about Randomness
   4. Godel and Ciphers?
   5. Characteristics of Computer-Based RNG's
   6. Number of Sequences
   7. Cycles
   8. Some Potential Attacks
   9. Inference Resistance by Exposure Class
   10. Information and Penetration
   11. Inference versus Prediction
   12. Cryptographic RNG Requirements
4. RNG Techniques
   1. Chaos
   2. Cebysev Mixing
   3. Cellular Automata
   4. $x^2 \bmod N$
   5. Linear Congruential
   6. Linear Feedback Shift Register
   7. Non-Linear Shift Register
   8. Clock-Controlled Shift Registers
   9. Generalized Feedback Shift Register
   10. Additive RNG
5. Randomizers and Isolators
   1. One-Way Functions
   2. Checksum Algorithms
   3. CRC's
   4. Randomizers
   5. Isolation Techniques
6. Other Randomness Techniques
   1. "Really Random" Values
   2. Cycle Detection
   3. Polynomial Degree
   4. Sequence Customization and Number of Terms
   5. Finding Primitive Mod 2 Polynomials
   6. Combined RNG's
   7. Random Permutations
7. RNG Exhaustive State Experiments
   1. Exhaustive State Analysis
   2. Results
   3. Discussion
8. Summary
9. Comments

Terry Ritter, his *current address*, and his *top page*.

Last updated: 1996-06-26

RNG Implementations: A Literature Survey

Research Comments from Ciphers By Ritter

Terry Ritter

Some of the well-known fully-worked-out RNG proposals with an obvious or explicit computer implementation.

Contents

- 1979
  - Bright and Enison describe what later becomes known as a GFSR, with sequence interruption for cryptographic use, and some of the tests used to evaluate the sequence.
  - Schrage describes how to evaluate a linear multiplicative generator *in* pieces, so that the computation range is not exceeded.
- 1982
  - Wichmann and Hill present one of the first portable designs (in FORTRAN). Based internally on three "byte-size" linear congruential generators, the three intermediate values never exceed 16-bit range.
- 1988
  - L'Ecuyer gives some new "Efficient and Portable" (and tested) RNG designs. These are "combined multiplicative linear congruential" designs for 16- and 32-bit wordlengths, in easy Pascal.
  - The Park and Miller design "Minimal Standard" RNG's. These are simple linear congruential designs for short wordlengths, also in easy Pascal.
- 1989
  - Press and Teukolsky of *Numerical Recipes* fame present a quasirandom sequence for improved Monte Carlo integration.
- 1990
  - Macomber and White provide a fully worked-out GFSR design (implemented in BASIC). Initialization will take a while.
  - Fushimi design a GFSR RNG, implement it in portable C, and test it. Again, initialization is a problem.
  - Ripley treats us to an insider's view of RNG's for simulation. There is also a new, faster (though still slow) approach to GFSR initialization.
- 1991
  - Tezuka and L'Ecuyer design an RNG (a combined pair of ordinary shift-register RNG's), implement it in portable C, and test it.
- 1992
  - Collins *et. al.* give a simple, mainly pedagogic, real-number RNG based on the logistic equation. This requires a simple transformation to produce the uniform distribution, and a different simple transformation gives a distribution which is very close to Gaussian.
  - Press and Teukolsky present the new RNG's from *Numerical Recipes* new edition. These are basically the Park-Miller "Minimal Standard" and L'Ecuyer "Efficient and Portable" designs, with added shuffling.
- 1993
  - Only five years later, Marsaglia responds to the Park-Miller "Minimal Standard" designs.
- 1994
  - Marsaglia and Zaman critique the new *Numerical Recipes* generators (Park-Miller "Minimal Standard" and L'Ecuyer "Efficient and Portable"). They particularly discuss ran2, and present mzran() (in Fortran) and mzran13() (in C) as alternatives.

1979 -- Bright and Enison

Bright, H. and R. Enison. 1979. Quasi-Random Number Sequences from a Long-Period TLP Generator with Remarks on Application to Cryptography. *Computing Surveys*. 11(4): 357-370.

"The background of the Lehmer linear congruential algorithms that are in almost universal use for random sequence generation is reviewed; the reasoning is given for choosing the somewhat different Tausworthe linear recurrence (mod 2) type, using the bit selection algorithm of Lewis and Payne; and the initial statistical evaluation is reported on. A review is given of the intuitive process by which the Tausworthe approach, with a generating primitive trinomial of Mersenne prime exponent degree, was chosen."

"We have developed a generator which uses the basic Tausworthe algorithm . . . ." "Lewis-Payne bit selection appears to be superior to decimation: To generate *r*-bit numbers, each of the *r* bits is chosen from a different part of the same Tausworthe sequence with a constant gap between bits."

"Our TLP generator uses the trinomial $x^{521} + x^{32} + 1$ to generate 64-bit numbers that are 8-distributed and have good *k*-distributivity for *k* > 8. The period of the sequence is $2^{521} - 1$. The LP gap is 52,100."

"Statistical Testing Criteria"

1. "*Equidistribution or Frequency Test*. One counts the number of times a member of the sequence falls into a given interval. The number should be approximately the same for intervals of the same length if the sequence is uniform."
2. "*Serial Test*. This is a higher dimensional version of the equidistribution test. One counts the number of times a *k*-tuple . . . of members of the sequence falls into a given *k*-dimensional cell. If this test is passed, the sequence is said to be *k*-distributed." "Other tests of *k*-distributivity are
   1. "Poker test: One considers groups of *k* members of the sequence and counts the number of distinct values represented in each group."
   2. "Maximum (minimum) of *k*: One plots the distribution of the function max (min) [of a *k*-tuple]."
   3. "Sum of *k*"
3. "*Gap Test*. One plots the distribution of gaps in the sequence of various lengths. . . ."
4. "*Runs Test*. One plots the distribution of maximal ascending (descending) runs of various lengths."
5. "*Coupon Collector's Test*. One chooses a suitably small integer *d* and divides the universe into *d* intervals. Then each member of the sequence falls into one such interval. One plots the distribution of runs of various lengths required to have all *d* intervals represented."
6. "*Permutation Test*. One studies the order relations between the members of the sequence in groups of *k*. Each of the *k*! possible orders should occur about equally often. If the universe is large, the probability of equality is small; otherwise, equal members may be disregarded."
7. "*Serial Correlation Test*. One computes the correlation coefficient between consecutive members of the sequence. This gives the serial correlation for lag 1. Similarly, one may get the serial correlation for lag *k* by computing the correlation coefficient between x_i and x_{i+k}. This is to show that the members of the sequence are independent."

"Other tests have been proposed and used. Lewis and Payne [Lewi73b] ran a conditional bit test, which tested the independence of each bit in a string from the others, as well as a Fourier transform test."

1979 -- Schrage

Schrage, L. 1979. A More Portable Fortran Random Number Generator. *ACM Transactions on Mathematical Software*. 5(2): 132-138.

"A Fortran implementation of a random number generator is described which produces a sequence of random integers that is machine independent as long as the machine can represent all integers in the interval {-2**31 + 1, 2**31 - 1}."

The innovation here is not the RNG, but the method of evaluation, which guarantees that the intermediate values of the generator remain in range.

1982 -- Wichmann and Hill

Wichmann, B. and I. Hill. 1982. Algorithm AS 183. An Efficient and Portable Pseudo-random Number Generator. *Applied Statistics*. 31: 188-190.

"We claim [our algorithm] is reasonably short, reasonably fast, machine-independent, easily programmed in any language, and statistically sound. It has a cycle length exceeding $[6.95 \times 10^{12}]$."

"The algorithm produces numbers rectangularly distributed between 0 and 1, excluding endpoints." (Actually, zero can occur due to rounding error, but a simple test suffices to tame the situation.)

1988 -- L'Ecuyer

L'Ecuyer, P. 1988. Efficient and Portable Combined Random Number Generators. *Communications of the ACM*. 31(6): 742-749, 774.

"**ABSTRACT:** In this paper we present an efficient way to combine two or more Multiplicative Linear Congruential Generators (MLCGs) and propose several new generators. The individual MLCGs, making up the proposed combined generators, satisfy stringent theoretical criteria for the quality of the sequence they produce (based on the Spectral Test) and are easy to implement in a portable way."

Internally, we have two smaller MLCG's each of which can be implemented easily on 32-bit machines. The internal generators are combined by subtraction to a positive value in the range of one of the generators, then converted to a uniform deviate by multiplication by 2^{-31}.

"For 32-bit computers, we suggest i=2, m1 = 2147483563, a1 = 40014, m2 = 2147483399 and a2 = 40692." " . . . the combined generator has period . . . 2.30584 x 10^{18}." "It works as long as the machine can represent all integers in the range [-2^{31}+85, 2^{31}-85]." "Notice that the function will never return 0.0 or 1.0, as long as REAL variables have at least 23-bit mantissa . . . ."

"We performed 21 different tests . . . ."

1. Equidistribution test, using chi-square, d=64, n=1000, N=10000.
2. Equidistribution test, using chi-square, d=256, n=10000, N=10000.
3. Serial test with pairs (2-dimensional), d=64, n=100000, N=10000.
4. Serial test with triplets (3-dimensional), d=16, n=100000, N=1000.
5. Serial test with quadruplets (4-dimensional), d=8, n=100000, N=1000.
6. Gap test, a=0, b=.05, t=15, n=10000, N=1000.
7. Gap test, a=.35, b=1, t=15, n=10000, N=1000.
8. Gap test, a=1/3, b=2/3, t=10, n=10000, N=1000.
9. Poker test, k=4, d=4, n=10000, N=1000.
10. Poker test, k=6, d=4, n=10000, N=1000.
11. Poker test, k=6, d=8, n=10000, N=1000.
12. Poker test, k=8, d=16, n=10000, N=1000.
13. Coupon's collector test, d=5, t=25, n=10000, N=1000.
14. Coupon's collector test, d=10, t=40, n=10000, N=1000.
15. Permutation test, t=3, n=10000, N=1000.
16. Permutation test, t=5, n=10000, N=1000.
17. Runs-up test, n=100000, N=1000.
18. Maximum-of-t test, t=8, d=128, n=10000, N=1000.
19. Collision test, 6 dimensions, d=8, n=20000, N=100.
20. Collision test, 10 dimensions, d=4, n=20000, N=100.
21. Collision test, 20 dimensions, d=2, n=20000, N=100.

1988 -- Park and Miller

Park, S. and K. Miller. 1988. Random Number Generators: Good Ones are Hard to Find. *Communications of the ACM*. 31(10): 1192-1201.

The so-called "Minimal Standard" RNG turns out to be a fairly important reference point, if not an actual standard.

" . . . the widespread adoption of good, portable, *industry standard* software for random number generation has proven to be an elusive goal. Many generators have been written, most of them have demonstrably non-random characteristics, and some are embarrassingly bad."

"To the non-specialist, the construction of a random number generator may appear to be the kind of thing that any good programmer can do easily. Over the years many programmers have unwittingly demonstrated that it is all too easy to 'hack' a procedure that will produce a strange looking, apparently unpredictable sequence of numbers. It is fundamentally more difficult, however, to write quality software which produces what is really desired -- a virtually infinite sequence of statistically *independent* random numbers, *uniformly distributed* between 0 and 1. This is a key point: strange and unpredictable is not necessarily random."

The selected design is a multiplicative linear congruential generator (MLCG):

$$z_{i+1} = 16807 z_i \bmod 2^{31} - 1.$$

There are two INTEGER and two REAL implementations in easy Pascal. The first two require 46-bit INTEGERS or REALs. Schrange's method is used in the second pair of designs to keep intermediate values within the 32-bit range.

1989 -- Press and Teukolsky

Press, W. and S. Teukolsky. 1989. Quasi- (that is, Sub-) Random Numbers. *Computers in Physics*. Nov/Dec. 76-79.

"Sometimes we want to pick sample points in an *n*-dimensional space that are spread out as uniformly as possible. The most obvious case is where we want to integrate a function over *n*-dimensional space by a Monte Carlo method."

"In the familiar case of *uncorrelated* points, e.g., those generated by a good pseudorandom number generator, the fractional error of the estimated integral will decrease by the equally familiar 1 / SQRT(N) law."

"A simple counterexample is to choose sample points that lie on a Cartesian grid, and to sample each grid point exactly once . . . . The Monte Carlo method thus becomes a deterministic quadrature scheme . . . whose fractional error decreases at least as fast as 1 / *N* . . . ."

"Sequences of *n*-tuples that fill *n* space more uniformly than uncorrelated random points are called *quasirandom* sequences."

Actual FORTRAN code is given for such a generator.

1990 -- Macomber and White

Macomber, J. and C. White. 1990. An *n*-Dimensional Uniform Random Number Generator Suitable for IBM-Compatible Microcomputers. *Interfaces*. 20(3): 49-59.

"[There is] a basic validity problem with mixed congruential RNG's in multidimensional situations [Marsaglia 1968]. Increasing the dimensionality of a hypercube tends to increase the lattice effect. Therefore, what a user believes to be a reliable RNG can become entirely predictable or obviously nonrandom in *n*-dimensional applications."

"An RNG that overcomes many of the shortcomings of congruential generators is the generalized feedback shift register generator, of GFSR generator [Kennedy 1975; Lewis 1975]."

"The GFSR RNG requires a starting array of randomized binary digits in order to work. To generate these starting numbers, a setup program is required to initialize the generator."

"With the incorporation of Tezuka's test [1987b] within this environment, global *n*-space uniformity can be guaranteed."

1990 -- Fushimi

Fushimi, M. 1990. Random number generation with the recursion
$X_t = X_{t-3p}$ xor X_{t-3q}. *Journal of Computational and Applied Mathematics*. 31: 195-118. North-Holland.

"*Abstract:* A generalized feedback shift register (GFSR) algorithm proposed by Lewis and Payne (1973) uses a primitive trinomial to generate a sequence of pseudorandom numbers. We propose a similar algorithm which uses a primitive polynomial with many non-zero terms, but generates a number as fast as the original GFSR algorithm. Our sequence is guaranteed to be equidistributed in higher dimensions and to have a good autocorrelation property."

"Marsaglia [17] pointed out about 20 years ago that the points in the unit *n*-cube generated by the Lehmer's linear congruential method lie in a relatively small number of parallel hyperplanes. All sequences generated with recursions of low degree over the finite field have the flaw of sparseness in *n*-space . . . ."

"We have developed a new method of generating pseudorandom numbers by linear recurrence modulo two. The method uses the recurrence formula

$$X_t = X_{t-3p} \text{ xor } X_{t-3q}$$

instead of the formula

$$X_t = X_{t-3p} \text{ xor } X_{t-q}$$

proposed by Lewis and Payne, where $1 + \sigma^q + \sigma^p$ is a primitive polynomial of degree *p* over the Galois field GF(2) . . . ." "The crucial point in using this generator . . . is the initialization procedure."

"Using the primitive trinomial $1 + z^{32} + z^{521}$, we have performed extensive statistical tests on number sequences generated by our method."

1990 -- Ripley

Ripley, B. 1990. Thoughts on pseudorandom number generators. *Journal of Computational and Applied Mathematics*. 31: 153-163. North-Holland.

"*Abstract:* Much of the informal discussion at the Workshop concerned the merits of different pseudorandom number generators. Here we record some comments based on comparing generators across a wide range of machines."

"The idea properties of a good general-purpose pseudorandom number generator are easy to agree but impossible to achieve simultaneously. They include

- a very good approximation to uniform distribution,
- a very close to independent output in a moderate number of dimensions,
- repeatability from a simply specified starting point,
- speed,
- a very long period (at least 2^{50})."

"It is by now known that well-designed congruential generators have a *k*-dimensional output which fills a lattice in $[0,1]^k$. All of these examples have adequate lattice structure in up to 8 dimensions [13, Sect. 2.4] and so satisfy the first two requirements, at least weakly. It hardly seems worth searching for 'optimal' multipliers, since most choices are reasonably good. (I find the criteria of Fishman and Moore [3] unreasonably stringent. Generators whose criteria differ by a factor of two are for practical purposes indistinguishable.)"

"Some of these generators (notably **Marsg**) have a poor two-dimensional discrepancy." "I believe that discrepancy is not a good measure of a *pseudo*random number generator, and may not be a good measure for a *quasi*random sequence . . . ."

"The alternative family is that of *shift-register* generators . . . ." "The snag with GFSRs is their initialization." "FORTRAN code is given in the Appendix."

"The reader may already have guessed which generators I actually use. For all but the most demanding problems **Marsg** suffices. It is simple to program in assembler . . . and easy to check against a double-precision implementation. If any doubt arises from the results of the simulation experiment I use **GFSR521** as a cross-check, but have never had cause to doubt **Marsg**. However, the period of **Marsg** is beginning to seem too small, and GFSRs have the edge for the future . . . ."

1990 -- Marsaglia and Zaman

Marsaglia, G. and A. Zaman. 1990. Toward a Universal Random Number Generator. *Statistics and Probability Letters*. 8: 35-39. North-Holland.

"*Abstract:* This article describes an approach toward a random number generator that passes all the stringent tests for randomness we have put to it, and that is able to reproduce exactly the same sequence of uniform random variables in a wide variety of computers, including TRS80, Apple, Macintosh, Commodore, Kaypro, IBM PC, AT, PC and AT clones, Sun, Vax, IBM 360/370, 3090, Amdahl, CDC Cyber and even 205 and ETA supercomputers."

This is a combined RNG. One part is a 97-element "lagged-Fibonacci" generator using multiplication. The other part is a linear congruential generator with modulus 169. This is a little more complex, and the write-up a little less clear, than other proposed "standard" designs. Includes Fortran code.

1991 -- Tezuka and L'Ecuyer

Tezuka, S. and P. L'Ecuyer. 1991. Efficient and Portable Combined Tausworthe Random Number Generators. *ACM Transactions on Modeling and Computer Simulation*. 1(2): 99-112.

"In this paper, we propose three combined Tausworthe random number generators with period length about 10^{18}, whose *k*-distribution properties are good and which can be implemented in a portable way. These generators are found through an exhaustive search for the combination with the best lattice structure in *GF(2,e)P*, the *k*-dimensional vector space over the field of all Laurent series with coefficients in *GF*(s). We then apply a battery of statistical tests to these generators for the comprehensive investigation of their empirical statistical properties. No apparent defect was found. In the appendix, we give a sample program in C for the generators."

1992 -- Collins, *et. al.*

Collins, J., *et. al.* 1992. A random number generator based on the logit transform of the logistic variable. *Computers in Physics*. 6(6): 630-632.

[Note that the term "nonperiodic" in the following paragraph refers to the theoretical situation of mathematical infinite-precision real numbers. In the context of computer floating-point with a limited-precision, no iterative RNG can possibly be "nonperiodic." In fact, we now have the worrisome possibility of multiple cycles of various lengths.]

"A nonperiodic random number generator, which is based on the logistic equation, is presented. A simple transformation that operates on the logistic variable and leads to a sequence of random numbers with a near-Gaussian distribution, is described and discussed. The associated algorithm can be easily utilized in laboratory exercises, classroom demonstrations, and software written for stochastic modeling purposes."

1992 -- Press and Teukolsky

Press, W. and S. Teukolsky. 1992. Portable Random Number Generators. *Computers in Physics*. 6(5): 522-524.

"Publication of [our] new editions gives us a unique opportunity for confession: There are several sections in the old books that are embarrassingly bad." "Worst among them, we think, is the section on generators of uniform random numbers . . . [we gave] two generators (RAN1 and RAN2) that are at best only mediocre."

"Our new editions, and this column, provide three much better exemplars, named ran0, ran1, and ran2. (Routines in the new editions are named in lower case, in part to distinguish them from old routines.)"

ran0 is the Park-Miller "Minimal Standard" design. ran1 is the same generator with Bays-Durham shuffling. ran2 is the L'Ecuyer "Efficient and Portable" combined design, with added shuffling. A FORTRAN listing for each is given.

1993 -- Marsaglia

Marsaglia, G. 1993. Remarks on Choosing and Implementing Random Number Generators. *Communications of the ACM*. 36(7): 105-108.

(Comments on Park-Miller "Minimal Standard" designs.)

- "The congruential generator $x_n = 16807x_{n-1} \bmod 2^{31} - 1$ is a good generator, but not a great generator."
- "The redeeming feature of the generator is the closeness of its modulus to a power of 2, providing nice, but tricky, machine language implementations. But for such implementations, 2^{32} - 2 or 2^{32} - 5 seem better choices."

"**Summary.** Generators using modulus 2^{31} - 1 are attractive only if they are implemented in machine language, but for such implementations the modulus 2^{32} - 2 is preferable

---

1994 -- Marsaglia and Zaman

Marsaglia, G. and A. Zaman. 1994. Some portable very-long period random number generators. *Computers in Physics*. 8(1): 117-121.

"It is found that a proposed random number generator ran2, recently presented in the *Numerical Recipes* column . . . is a good one, but a number of generators are presented that are at least as good and are simpler, much faster, and with periods "billions and billions" of times longer. They are presented not necessarily to supplant ran2, but to put it in perspective. Any serious user of Monte Carlo methods should have a variety of random number generators from which to choose. In addition to two specific programs, one in Fortran and one in C, a framework is offered within which the readers can easily fashion their own generators with periods ranging from 10^{27} - 10^{101}."

Terry Ritter, his current address, and his top page.

Last updated: 1996-06-30

Random Electrical Noise: A Literature Survey

Research Comments on Ciphers By Ritter

Terry Ritter

INTRODUCTION

Comments on the production, processing, detection and conversion to uniform distribution.

Producing Electronic Noise

If we wish to generate and use electrical noise, we have two main sources: Thermal and Shot. Both are fundamental sources of "white" noise, meaning that we have a deep statistical understanding of how these sources behave. Unfortunately, this may not be particularly useful if the noise we have is actually due to variable processing-related problems.

- Thermal or Johnson noise results from the Brownian motion of ionized molecules within a resistance. Thermal noise is entirely fundamental and cannot be eliminated (although the effect can be reduced by reducing or cooling the resistor).

  Carbon-composition resistors may give more noise than expected; this added noise is from device fabrication, varies widely in production, and is not necessarily "white." Thus, carbon-film or wirewound resistors are more satisfactory thermal noise sources.

  To verify that a noise source is indeed producing thermal noise, it may be useful to "short out" the source resistance and verify a marked reduction in resulting noise (hopefully to under 1/10 of the original value). This of course implies an ability to quantify the mean amplitude of the noise signal.

  Shot noise typically results from the flow of electrons through a highly-charged field like a vacuum tube or semiconductor junction. Ultimately, electron flow is the movement of discrete charges, and surrounding the mean flow rate is a distribution related to the launch time and momentum for individual charge carriers entering the charged field.

  Shot noise is fundamental, so no true zener can be noiseless, and any especially "noisy" zener must be producing something beyond shot noise. Since other noise sources (especially contact noise) are typically unrelated to device fabrication and are not necessarily "white," this "extra" noise should be avoidable. We should thus seek the lowest-noise zener for noise sources. Since zener noise levels will vary with temperature, noise form of automatic gain control (AGC) may be necessary.

  Shot noise in semiconductor junctions varies inversely as the square root of the current, but for small currents other difficulties arise, and, again, the resulting noise is not necessarily "white." To verify that a semiconductor junction is indeed producing shot noise, it may be helpful to increase the current by 100x and measure the resulting noise at 1/10 the original value. This of course implies an ability to quantify the mean amplitude of the produced noise signal.

The various other properties we might measure — such as the time between zeros (or any other level) does not seem to give us any particular distribution advantage. Possibly we could show that "any" sort of noise is sufficient for some sort of sampling to produce one uniformly distributed bit, but this has not been established.

Post-Production Analog Processing

It will be necessary to greatly amplify the noise in a linear broadband manner. This is harder than it sounds, because common self-compensated op-amps will have a 6 dB/octave roll-off for stability, and we need the full total amplitude bandwidth. Furthermore, if the bandwidth will define the width of the minimum pulse and the maximum rate at which the noise can be sampled.) Producing ideal noise from a fundamental source is of little help if we modify the result prior to detection.

The noise output from zener diodes tends to vary through time and especially temperature. Even with thermal noise, some form of automatic gain control is probably necessary over production and time, and will imply some amount of short-term amplitude correlation.

Detection and Conversion to Uniform Distribution

If we have a white noise source, we can assume that it will have a Gaussian distribution of instantaneous amplitude. Thus, if we sample the analog amplitude at random, we can expect to get random values in Gaussian distribution. This will require a very fast analog sample-and-hold system, and will — at best — produce results in the particular form of Gaussian distribution which the machine develops. This is not a general system.

Suppose we simply detection amplitude so that the noise signal is above detection exactly half the time (this probably implies some sort of automatic control, which also implies a small amount of short-term amplitude correlation). If we then sample at random times (with a wide random period between samples), we can produce one uniformly-distributed bit per sample. However, it is difficult to guarantee that any repetitive process will sample "at random times." Adjacent close samples may support unwanted correlations.

If we can detect individual noise pulses, we can assume that the number of pulses which arrive in a particular time is Poisson distributed. By increasing the detection amplitude until few pulses are detected we can increase the probability that the pulses we do detect will be independent. Unfortunately, this "trick" will always be some pulses so close together to distinguish as two pulses rather than one, and we must reduce this effect with wide bandwidth, high-speed detection, and large counts.) Then we compare the parity of the current count to get one (almost) uniformly distributed random bit. We note that this method does require random sampling times, something difficult to require of a repetitive machine.

To verify correct operation of the noise source we might collect and verify either or both Gaussian amplitude and Poisson pulse-count distributions during normal operation. (This is in addition to some hardware check to verify that the detected noise is produced by the expected source.)

Contents

- 1944
  - Rice gives the theoretical formulas.
- 1948
  - MacDonald describes the Brownian nature of thermal or Johnson noise.
- 1955
  - Burgess discusses shot and avalanche noise.
- 1956
  - Pierce discusses electrical noise in general, and thermal noise in particular.
- 1962
  - Ragazzini and Chang discuss the assumed Gaussian nature of noise.
- 1965
  - Oliver discusses statistics related to both thermal and shot noise.
- 1968
  - Haitz and Vollmer report results from measuring semiconductor avalanche noise at microwave frequencies.
- 1971
  - Johnson (the discoverer of thermal noise) describes thermal noise.
- 1976
  - Ott describes both thermal and shot noise, and practical noise measurement.
- 1983
  - Vergers describes shot noise in p-n junctions.
- 1989
  - Horowitz and Hill describe zener noise, thermal noise, shot noise, and noise measurement.

1944 -- Rice

Rice, S. 1944. Mathematical Analysis of Random Noise. *Bell System Technical Journal.* 23: 282-332; 24: 46-156.

Expected Zeros Per Second

"For an ideal band-pass filter whose pass band extends from f_1 to f_2 the expected number of zeros per second is"

$$2\left(1/3\left(f_2^2 - f_1^2\right)/\left(f_2 - f_1\right)\right)^{1/2}$$

"When f_1 is zero this becomes 1.155 f_2, and when f_1 is very nearly equal to f_2 it approaches $f_2 + f_2$."

Expected Maxima Per Second

"For a band-pass filter the expected number of maxima per second is"

$$\left[3/5\left(f_2^5 - f_1^5\right)/\left(f_2^3 - f_1^3\right)\right]^{1/2}$$

"For a low-pass filter where $f_1 = 0$ this number is 0.775 f_2."

"The expected number of maxima per second lying above the line $X(t) = \lambda$ is approximately, when λ is large,"

$$e^{-\lambda^2/2\,\psi_0 0}$$

"where $\psi_0 0$ is the mean square value of $X(t)$."

1948 -- MacDonald

MacDonald, D. 1948. The Brownian Movement and Spontaneous Fluctuations of Electricity. *Res. Appl. in Industry.* 1: 194-203.
Also reprinted in: *Electrical Noise: Fundamentals and Sources.* 1977. M. Gupta, Ed. IEEE Press. 7-16.

"In 1827 the biologist Robert Brown was studying under his microscope the pollen grains, some 0.0002 inch in length, of the plant *Clarchia pulchella.*

"While examining the form of these particles immersed in water, I observed many of them very evidently in motion. These motions were such as to satisfy me . . . that they arose neither from currents in the fluid, nor from its gradual evaporation, but belonged to the particle itself.

". . . It was left for the elder Einstein, apparently fundamental, and A. Einstein, in a series of classical papers, was the first to provide a clear analysis of the problem as arising from continuous and random molecular bombardments . . . ."

". . . von Smolan in 1879 had considered the possibility of molecular bombardment but had concluded because the impulse due to one collision was so minute, that this could not be the cause; for he opined that since all directions in space are equally likely the cumulative effect of many random collisions could only be of the same magnitude. The error is a common one and arises essentially from implicitly regarding a random process as made up of regularly alternating favourable and unfavourable events; such a process is, however, a highly ordered one and it is those very 'runs' of favourable (or unfavourable) events, which we sometimes regard as against the 'laws of chance', which characterize a random process and give rise to the relatively large fluctuations observed."

"The essentials are well illustrated by the fundamental 'random walk' problem, as first posed by Karl Pearson in 1905. A man (personality very drunk) takes steps of equal length, l, from a starting point O on after the other in successively random directions. Where is he likely to be after n steps? Lord Rayleigh answered the problem immediately where n is large; the probability that he is at a distance between r and r+dr from his starting point is

$$p(r)dr = (2r/nl^2)e^{-(r/nl)2}\,dr.$$

"His average distance is therefore *[the integral from 0 to infinity of r * p(r) dr, or]*

$$\left(\pi n l^2/2\right)^{1/2}$$

"and thus increases with the square root of the time for which he continues the walk. . . . "

[It appears that this discussion of the Rayleigh distribution was introduced solely to counter the argument that random microscopic forces could not sum to produce a macroscopic effect. There appears to be no implication that electrical noise itself should have a Rayleigh distribution.]

1955 -- Burgess

Burgess, R. 1955. Electrical fluctuations in semiconductors. *British J. Appl. Phys.* 6: 185-190.
Also reprinted in: *Electrical Noise: Fundamentals and Sources.* 1977. M. Gupta, Ed. IEEE Press. 59-64.

SHOT NOISE

"The term 'shot noise' was originally applied to the fluctuations of current in a saturated vacuum diode due to the randomness of electron emission from the cathode." At low frequencies such that the electron transit time t is small compared with 1/w, the *[Fourier] transform F(f) = e* and the spectral density assumes the simple form 2eI. The concept of randomness of rate of emission implies that the process is determined by a stationary Poisson distribution."

"Another important instance of shot noise arises in the other extreme from a uniform semiconductor, namely the motion of carriers across a high-field transition region, e.g. at a metallic contact or at a p-n junction. Normally the carriers across such a region would be of the order of 10^7 c/m/s and the width of the region would be in the range of 10^{-5} to 10^{-6} cm so that the transit time would be negligible except at the highest microwave frequencies. Furthermore it may be readily shown that since the change in quasi-Fermi level for the carriers across the transition region is very nearly equal to the applied voltage, the effect of each electron transit is effectively to induce a current impulse e$\delta(t)$, and thus full shot noise may be attributed to the flow."

AVALANCHE NOISE

"When a barrier region is subjected to reverse bias the electric field may reach the order of 10^5 V/cm or greater, and at these fields there occur phenomena which cause a rapid increase of current and eventual breakdown; it has furthermore been observed that the current is 'noisy' in this region, increasingly impulsive as breakdown is approached."

"In silicon junctions McKay[14] observed that at the onset of breakdown there appears a distinction form of impulsive noise consisting of a random sequence of rectangular current pulses of variable duration but constant amplitude." It is possible that the inevitable inhomogeneity of the semiconductor in the neighbourhood of the junction gives rise to small regions (or 'weak spots') in which breakdown occurs for lower applied voltage than elsewhere and this localized breakdown will switch from an 'off' to an 'on' condition and back again, triggered by random fluctuation."

1956 -- Pierce

Pierce, J. 1956. Physical Sources of Noise. *Proc. IRE.* 44: 601-608.
Also reprinted in: *Electrical Noise: Fundamentals and Sources.* 1977. M. Gupta, Ed. IEEE Press. 51-58.

"Many sorts of electrical signals are called noise." ". . . many engineers have come to regard any interfering signal of a more or less unpredictable nature as noise." "The theory of noise presented here is not valid for all signals or phenomena which the engineer may identify as noise."

"The theory of noise is best adapted to handling signals which originate in truly random processes, such as the emission of electrons from a photo-surface or a hot cathode, or the thermal agitation of charges in a resistor. When a cathode emits electrons at so slow a rate that we observe their effects in a circuit as separate pulses, we have *impulse noise,* and the theory of noise has something to say about this. When electrons are randomly emitted so rapidly that the pulses they produce in the circuit overlap, the statistics of large numbers applies, and the theory of noise tells us a great deal that must be true of a large class of noise signals, despite differences in the exact nature of their sources."

Johnson Noise

"The first source of noise which we consider is Johnson noise, the thermal noise from a resistor. The engineering fact is that a resistor of resistance R acts like a noise generator."

"What is the source of Johnson noise? In an ordinary resistor, it is a summation of the effects of the very short current pulses of many electrons as they travel between collisions, each pulse individually having a flat spectrum. In this case the noise is a manifestation of the Brownian movement of the electrons in the resistor."

1962 -- Ragazzini and Chang

Ragazzini, J and S. Chang. 1962. Noise and Random Processes. *Proc. IRE.* 50: 1146-1151.
Also reprinted in: *Electrical Noise: Fundamentals and Sources.* 1977. M. Gupta, Ed. IEEE Press. 25-30.

"In most systems which are of interest to the engineer and the designer, random processes are assumed to be Gaussian or at least assumed to be satisfactorily approximated by such a distribution." ". . . a random signal or noise is usually generated by a large number of independent events . . . " "It follows from the central limit theorem in probability that . . . the amplitude of such a signal *f is f normally distributed . . . "*

1965 -- Oliver

Oliver, B. 1965. Thermal and Quantum Noise. *Proc. IEE.* 53: 436-454.
Also reprinted in: *Electrical Noise: Fundamentals and Sources.* 1977. M. Gupta, Ed. IEEE Press. 129-148.

Statistics of Thermal Noise

". . . any linearly filtered or amplified thermal noise wave has a Gaussian distribution of instantaneous amplitude."

"Usually the *Gaussian* amplitude distribution of thermal noise is developed on the basis of a model source containing a very large number of independent generators each of which produces an infinitesimal contribution to the resultant amplitude. For example, in a resistor, each conduction band electron acts as it buffeted about produces a random current wave. The total current is then shown to have a Gaussian distribution by the Central Limit Theorem."

". . . the envelope, A(t), *[that is, the peak amplitude]* of any thermal noise wave has a Rayleigh distribution . . . "

Shot Noise

"Whenever discrete particles arrive at random times there will be fluctuations in the rate of arrival. It is these fluctuations that constitute shot noise."

"Let us assume that a particle is equally likely to arrive at any time, and that the average rate of arrival is r." "Under these conditions the numbers of arrivals in a given length of time are distributed according to the well-known Poisson distribution."

". . . when the average number of arrivals during the observing time is large, the fluctuations will approach a Gaussian distribution about the mean with sigma = $a^{1/2}$."

1968 -- Haitz and Voltmer

Haitz, R and F. Voltmer. 1968. Noise of a Self-Sustaining Avalanche Discharge in Silicon: Studies at Microwave Frequencies. *J. Appl. Phys.* 39: 3379-3384.
Also reprinted in: *Electrical Noise: Fundamentals and Sources.* 1977. M. Gupta, Ed. IEEE Press. 327-332.

"The studies of avalanche noise reported by Haitz are extended to frequencies up to and above the avalanche frequency w_a. It is found that the open-circuit spectral voltage density is flat within +/- 5% from less than 100 Hz up to frequencies approaching w_a."

"Having the studies of low-frequency avalanche noise in mind it became evident that special precautions have to be taken in order to prevent the generation of excessive noise resulting from nonuniform breakdown. The combination of both a guard ring to prevent edge breakdown and a small breakdown area to reduce material nonuniformities were found to be satisfactory results."

"At currents below 2.5 mA the measured noise is larger than the noise predicted . . . This discrepancy, which is typical for avalanche diodes at low current densities, is not serious. It is caused by extremely small nonuniformities of the breakdown current."

Fig. 2 is a graph of "Open Circuit Spectral Voltage Density" in nV/SQRT(Hz) versus "Current" in mA. At 1 mA the graph shows about 100 nV/rt, at 10 mA about 40, and at 0.1 mA theory indicates about 400, but the actual device shows much more. For currents of 2.5 - 20 mA, theoretical and experimental avalanche noise this decrease with increased current. Across the graph range, theoretical noise appears to decrease by a factor of 10 with a current increase of 100.

1971 -- Johnson

Johnson, J. B. 1971. Electronic Noise: the first two decades. *IEEE Spectrum.* 8: 42-46.
Also reprinted in: *Electrical Noise: Fundamentals and Sources.* 1977. M. Gupta, Ed. IEEE Press. 17-21.

"In the 1918 paper, Dr. Schottky evidently assumes that the grosser current fluctuations produced by faulty tube structures . . . have been, or can be, eliminated, and he is left with two sources of noise that are of a much more fundamental nature. One he calls the 'Warmeeffekt,' in English now commonly named 'thermal noise.' This is a fluctuating voltage generated by electrical current flowing through a resistance in the input circuit of an amplifier, the other is the 'shroteffekt.' The motion of charge is a spontaneous and random flow of the electrical charge in the conductor in response to heat motion in its molecules."

"In the case of the 'thermal noise' . . . "the electric charge is in effect bold in long bags with walls relatively impervious to electrons at low temperature. The mass transport of charge along the bag, or wires, under the influence of heat motion, sets up the potential differences that generate the fluctuating output of the amplifier."

"When now one end of the conductor, the 'cathode' of the tube, is heated to incandescence, electrons can be emitted from the cathode surface to travel across the vacuum toward the anode. The electrons are emitted at random times, independent of each other, and they travel at different velocities, depending on initial velocity and voltage distribution for electron passage. In the case of a small electron emission, a small nearly steady flow of current is produced, with a superimposed smaller alternating current whose amplitude can be calculated from statistical theory. This small current flowing through the amplifier generates the 'Schroteffekt,' or shot effect, in the amplifier."

". . . for frequencies above certain values, the noise power is constant up to very high frequencies. For thermal noise this constant power extends also to low values, while for shot noise there are many exceptions and variations."

1976 -- Ott

Ott, H. 1976. *Noise Reduction Techniques in Electronic Systems.* John Wiley & Sons.

THERMAL NOISE

"Thermal noise comes from thermal agitation of electrons within a resistance, and it sets a lower limit on the noise present in a circuit. Thermal noise is also referred to as resistance noise or 'Johnson noise' (for J. B. Johnson, its discoverer.)" "He showed that the open-circuit rms noise voltage produced by a resistance is

$$V_T = 4kTRB\ V^2$$

where, k = Boltzmann's constant (1.38 x 10^{-23} joules/deg. Kelvin), T = Absolute temperature (deg. Kelvin), B = Noise bandwidth (Hz), R = Resistance (Ohms)." (p. 198)

"Although the rms value for thermal noise is well defined, the instantaneous value can only be defined in terms of probability. The instantaneous amplitude of thermal noise has a Gaussian, or normal, distribution." (p. 203)

"The crest factor of a waveform is defined as the ratio of the peak to the rms value." " . . . a crest value of approximately 4 is used for thermal noise." (p. 204)

SHOT NOISE

"Shot noise is associated with current flow across a potential barrier. It is due to the fluctuation of current around an average value resulting from the random emission of electrons (or holes). This noise is present in both vacuum tubes and semiconductors. In vacuum tubes, shot noise comes from the random emission of electrons from the cathode. In semiconductors, shot noise is due to random diffusion of carriers through the base of a transistor and the random generation and recombination of hole electron pairs."

"The shot effect was analyzed theoretically by W. Schottky in 1918. He showed that the rms noise current was equal to:

$$I_{sh} = 2qI_{dc}B\ )^{1/2}$$

where q = Electron charge (1.6 x 10^{-19} coulombs), I_{dc} = Average dc current (A), B = Noise bandwidth (Hz)."

"The power density for shot noise is constant with frequency and the amplitude has a Gaussian distribution. The noise is white noise and has the same characteristic as previously described for thermal noise."

" . . . by measuring the dc current through the device, the amount of noise can be very accurately determined." "A diode can be used as a white noise source. If shot noise is the predominant noise source in the diode, the rms value of the noise current can be determined simply by measuring the dc current through the diode." (pp. 208-209)

[This last statement appears false, given the experimental results in Haitz and Voltmer, the theory described in Vergers, and the noise graph given in Horowitz and Hill.]

MEASURING RANDOM NOISE

"An oscilloscope is an often overlooked, but excellent device for measuring white noise." "The rms value of white noise is approximately equal to the peak-to-peak value taken from the oscilloscope, divided by eight. When determining the peak-to-peak value on the oscilloscope, one or two peaks that are considerably greater than the rest of the waveform should be ignored. With a little experience, rms values can be accurately determined by this method." (p. 212)

1987 -- Vergers

Vergers, C. 1987. *Handbook of Electrical Noise.* TAB Books, Blue Ridge Summit, PA.

SHOT NOISE

"The term 'shot noise' arose from the study of random variations in the emission of electrons from the cathode of a vacuum tube. If these variations are amplified and listened to with a pair of headphones or a loudspeaker, they sound like 'lead shot' hitting a concrete wall. Shot noise has a flat spectral density like thermal noise. Therefore, shot noise can be considered a 'white noise' process." (p. 96)

Shot Noise in PN Junctions

"The shot noise generated in a pn junction has the same mathematical form as that of the temperature limited vacuum diode. The noise seems to be generated by a noise current generator in parallel with the dynamic resistance of the diode." (p. 108)

$$I_{sn} = (2eI_{dc}B)^{1/2}$$

where e = electron charge (1.6 x 10^{-19} coulombs), I = current in amperes, and B = bandwidth in Hertz.

"Likewise we may determine the shot noise voltage by applying Ohms Law." *[E = IR]*

$$E_{sn} = (2eI_{dc}Br_d^2)^{1/2}$$

where r_d is the dynamic resistance of the junction." (p. 109)

"From electronic physics it is known that the dynamic resistance of a pn junction depends on temperature and the direct current flowing through the junction. The dynamic resistance represents the ratio of a small change in diode voltage to a corresponding change in diode current."

$$r_d = kT/eI_{dc}$$

[we substitute and get]

$$E_{sn} = (2kTIB_{dc})^{1/2}$$

[we can even substitute again and get]

$$[\ E_{sn} = kT\ (\ 2\ /\ e)_{\ 1/2}\ ]$$

where k = Boltzmanns constant (1.38 x 10^{-23} Joules/deg. Kelvin), T = Temperature in degrees Kelvin, B = Bandwidth in Hertz, r_d = dynamic resistance, and e = Electron charge (1.6 x 10^{-19} coulombs).

"There is a rather interesting relation between the shot noise voltage across the junction and the dynamic resistance r_d. Since r_d is inversely proportional to direct current, the dynamic resistance falls as direct current increases. This means that the shot noise voltage across the junction is decreased." " . . . shot noise current is proportional to the square root of direct current where dynamic resistance is inversely proportional to direct current. When direct current increases, dynamic resistance falls more quickly than shot noise current rises. The result is that the shot noise voltage becomes inversely related to' *[the square root of]* "direct current." (p. 110)

1989 -- Horowitz and Hill

Horowitz, P. and W. Hill. 1989. *The Art of Electronics.* 2nd Ed. Cambridge University Press.

6.14 Zener Diodes

"Zener diodes can be very noisy, and some IC suppliers offer them for sale under the name of noise diodes." "It may seem a true zener, make sure it has response at the frequencies you are measuring . . . . " "True rms meters also specify a 'crest factor' . . . " "For Gaussian noise, a crest factor of 3 to 5 is adequate."

"You can use a simple *average* rectifier and scale the reading by 1.11 to measure rms." "To get the rms voltage of Gaussian noise, multiply the 'rms' value you read on an averaging ac voltmeter by 1.13 (or add 1dB)."

"A third method. . . . consists of looking at the noise waveform on an oscilloscope: The rms voltage is 1/6 to 1/8 of the peak-to-peak reading . . . . It isn't very accurate, but at least there's no problem getting enough measurement bandwidth." (p. 454)

Terry Ritter, his current address, and his top page.

Last updated: 1996-08-15

Random Number Bibliography: A Literature Survey

Research Comments from Ciphers By Ritter

Terry Ritter

- 1995 - Rand Corporation
- 1995 - Bellcore
- 1995 - Agnew
- 1995 - Vorstum
- 1995 - Gude
- 1995 - Tsuchiya
- 1995 - Tsuchiya and Takagi
- 1995 - Takagi
- 1995 - Takagi and Tsuchiya
- 1995 - Bengio and Brassard
- 1995 - Michael
- 1995 - Vitale
- 1995 - Vitale, Wolfe and Chua
- 1995 - Gude
- 1995 - Gude and Tsuchiya
- 1995 - Tsuchiya
- 1995 - Tsuchiya and Takagi
- 1995 - Agnew
- 1995 - Agnew, Mullin, Rothaghe, Vanstone, Fuji-Hara and Glässer
- 1995 - Mullin, Rothaghe, Vanstone, Fuji-Hara and Glässer
- 1995 - Gude
- 1995 - Gude
- 1995 - Gude
- 1995 - Davis, Ruben and Frenzenburger
- 1995 - Davis, Ruben and Frenzenburger
- 1995 - Davis, Ruben and Frenzenburger

Randomness Tests: A Literature Survey

Research Comments from Ciphers By Ritter

Terry Ritter

I take a "random" *value* to be an arbitrary selection from among some defined set. I take a "random" *string* to be just a sequence of random values. Among other things, this means that a long sequence of zeros (or any other particular sequence) is just *unlikely*, not non-random. Unless, of course, such a sequence regularly turns up more often than one would expect.

When we are talk about "nonrandom" strings, we discuss the *distribution* of encountered strings. And, even if we have a perfectly random source, we cannot expect to see a "flat" distribution of actual random strings: Since we effectively sample with replacement, we must expect to find many duplicates before getting even one example of every possible string.

What we would like is some test which will indicate whether some accumulation of sequences is "random." But the best that any test can offer is the *probability* of getting such an accumulation under whatever assumptions we have. That is, there is always a valid *possibility*, no matter how small, that even very peculiar accumulations of strings could have occurred by chance. And this means that it is very important to run our tests multiple times.

When a "random" sequence is taken to be "without any shorter construction" we have a problem: Because it is impossible to check every possible construction, it is also impossible to say that a sequence could not be constructed in a shorter way (unless we restrict ourselves to some particular construction). But this is a fine analogy to cryptography itself: Because it is impossible to try every attack, it is also impossible to state that there is not some simpler attack which might work.

Contents

- 1963
  - von Neumann discusses randomness.
- 1965
  - *Theory:* Kolmogorov relates the randomness of a sequence to the shortest algorithm which can reproduce that sequence.
- 1971
  - *Algorithm:* Kak applies Walsh-Fourier transforms to measuring the amount of randomness in a finite random sequence.
- 1972
  - *Algorithm:* Phillips gives an algorithm for computing the uncertainty in a binary sequence.
  - *Algorithm:* Phillips gives an algorithm for computing the autocorrelation of a binary sequence.
- 1975
  - *Theory:* Chaitin describes the idea that randomness is related to the shortest function which can create a given sequence.
- 1976
  - *Text:* Bennett.
- 1977
  - *Algorithm:* Yuen proposes using Walsh transforms to test random number sequences.
- 1980
  - *Practice:* Atkinson reminds us that some tests are useful only when testing particular RNG designs.
- 1981
  - *Text:* Knuth is the standard text in the field.
- 1983
  - *Algorithm:* Hopkins gives an algorithm for the spectral test.
- 1985
  - *Theory and Tests:* Marsaglia discusses RNG construction and testing.
  - *Tests:* Beker and Piper give the conventional tests.
  - *Theory and Algorithm:* Blumer *et. al.* give a construction for searching text.
- 1987
  - *Test:* Rueppel gives us the linear complexity profile.
  - *Theory:* Tezuka analyzes GFSR sequences.
  - *Theory and Algorithm:* Blumer *et. al.* gives a data structure for inverted files.
  - *Test:* Feldman gives a randomness test based on a fast Walsh-Hadamard transform (FWT).
- 1988
  - *Theory:* Wanders analyzes Golomb's randomness postulates.
- 1989
  - *Theory:* Maurer and Massey provable cryptographic security for pseudorandom sequences.
  - *Test:* Richards describes the graphic display of a sequence for visual randomness inspection.
  - *Test:* Carroll describes a binary derivative test.
  - *Theory:* Beth and Dai relate Turing-Kolmogorov-Chaitin complexity and Linear Complexity.
  - *Theory:* Jansen shows how to find the shortest (possibly nonlinear) feedback shift register. Presumably this supplants Berlekamp-Massey for linear complexity measurements.
- 1990
  - *Test:* Feldman gives a spectral test based on Fourier transforms.
  - *Test:* Maurer develops the universal test.
  - *Theory:* Jansen and Boekee discuss the Directed Acyclic Word Graph (DAWG).
  - *Theory:* Jansen and Boekee give an interesting Ziv-Lempel approach to the production of sequences
- 1991
  - *Theory:* Compagner discusses correlation hierarchies.
  - *Test:* Mund introduces Ziv-Lempel complexity.
  - *Theory:* Compagner discusses randomness.
- 1992
  - *Tests:* L'Ecuyer surveys RNG testing.
- 1993
  - *Tests:* Marsaglia and Zaman give other RNG tests.
  - *Theory:* Maclaren discusses the different RNG requirements of cryptography and statistics.
- 1995
  - *Test:* Gustafson, Dawson and Golic approach the testing of large subsets.

1963 -- von Neumann

von Neumann, J. 1963. Various Techniques Used in Connection With Random Digits. *John mon Neumann, Collected Works.* A.H. Taub, ed., MacMillan.

". . . in tossing a coin it is probably easier to make two consecutive tosses independent than to toss heads with probability exactly one-half. If independence of successive tosses is assumed, we can reconstruct a 50-50 chance out of even a badly biased coin by tossing twice. If we get heads-heads or tails-tails, we reject the tosses and try again. If we get heads-tails (or tails-heads), we accept the result as heads (or tails)."

"Any one who considers arithmetical methods of producing random digits is, of course, in a state of sin. For, as has been pointed out several times, there is no such thing as a random number -- there are only methods to produce random numbers, and a strict arithmetic procedure of course is not such a method."

"If one nevertheless considers certain arithmetic methods in detail, it is quickly found that the critical thing about them is the behavior of round-off errors in mathematics."

1965 -- Kolmogorov

Kolmogorov, A. 1965. Three Approaches to the Quantitative Definition of Information. *Problems of Information Transmission.* 1: 1-17.

"There are two common approaches to the quantitative definition of 'information': combinatorial and probabilistic. The author briefly describes the major features of these approaches and introduces a new algorithmic approach that uses the theory of recursive functions."

1971 -- Kak

Kak, S. 1971. Classification of Random Binary Sequences Using Walsh-Fourier Analysis. *Proceedings of Applications of Walsh Functions.* 74-77. Washington, D.C., 1971.

"This paper presents a straightforward procedure using Walsh functions to determine the pattern in a binary sequence."

". . . classification of data amounts to the computation of structure with respect to some criterion."

"A sequence shall be said to have no pattern or be random if the number of independent amplitudes in the Wash-Fourier transform is equal to the length of the sequence itself, i.e., 2^k."

"The measure of randomness r(s) shall be defined by"

$$r(s) = \text{no. of independent amplitudes of } W(s) \,/\, \text{length of the sequence} = i(s) / L(s)$$

"The number of independent amplitudes of W(s) shall equal the number of its component Walsh waves." *[The number of non-zero terms?]*

1972 -- Phillips

Phillips, J. 1972. Algorithm AS 48. Uncertainty Function for a Binary Sequence. *Applied Statistics.* 21: 97-99.

"The purpose of the algorithm is to input . . . a sequence of symbols which are of two kinds . . . , to tabulate the frequency for all subsequences of all lengths up to a maximum limit *maxlength*, and compute, for each length *m*, the uncertainty function . . . ."

"The algorithm works by setting up a table of frequencies of all subsequences of the maximum length, each possible subsequence being treated as an implicit binary integer of *maxlength* digits . . . ." "Then repeatedly, the leftmost bit of the current integer is deleted, the integer is left-shifted one place, the next symbol input, coded as 0 or 1, added to the integer, and the frequency of the pattern represented by the result increased by 1." "The computation of the uncertainty functions from the tables is trivial."

1972 -- Phillips

Phillips, J. 1972. Algorithm AS 49. Autocorrelation Function for a Binary Sequence. *Applied Statistics.* 21: 100-103.

"The purpose of the algorithm is to input . . . a sequence of symbols, which are of two kinds . . . , to tabulate the frequencies with which symbols of each kink are followed at a given lag by symbols of either kind, this for all lags up to a given maximum *maxlag*, and to compute the [phi] correlation coefficient for each lag *m* . . . ."

1975 -- Chaitin

Chaitin, G. 1975. Randomness and Mathematical Proof. *Scientific American.* 232(5): 47-52.

"Tossing a coin 20 times can produce any one of 2^{20} (or a little more than a million) binary series, and each of them has exactly the same probability. Thus it should be no more surprising to obtain the series with an obvious pattern than to obtain one that seems to be random . . . . If origin in a probabilistic event were made the sole criterion of randomness, then both series would have to be considered random, and indeed so would all others . . . . The conclusion is singularly unhelpful in distinguishing the random from the orderly."

". . . 'incompressibility' is a property of all random numbers; indeed, we can proceed directly to define randomness in terms of incompressibility: A series of numbers is random if the smallest algorithm capable of specifying it to a computer has about the same number of bits of information as the series itself."

1976 -- Bennett

Bennett, W. 1976. *Scientific and Engineering Problem-Solving with The Computer.* Prentice-Hall.

Interesting approaches to Language correlations (Chapter 4) and Random Processes (Chapter 6).

1977 -- Yuen

Yuen, C. 1977. Testing Random Number Generators by Walsh Transform. *IEEE Transactions on Computers.* C-26(4): 329-333.

"*Abstract* -- A truly random sequence of numbers has an asymptotically flat Walsh power spectrum. This fact is used to devise a new test for the randomness of the output of random number generators."

"One essential randomness test is that of uncorrelatedness, i.e., that the autocorrelation of the sequence is approximately a *d*-function."

"A property equivalent to uncorrelatedness is that the power spectrum be flat."

"In this paper we propose another randomness test equivalent to the correlation test: that the *Walsh* power spectrum be flat." "Thus, testing the flatness of the spectrum is equivalent to testing for uncorrelatedness of the values of *x*."

". . . the band spectrum estimate can also be evaluated by spectrum averaging . . . ."

"With segment averaging there is no longer any difficulty with core requirements. When we wish to test a sequence of 2^n values, we would read in, or generate 2^{n-m} values at a time, compute the 2^{n-m}-point fast Walsh transform of the segment, square, and add the squares to the 2^{n-m} memory locations which have been initially set to zero. After all 2^m segments have been processed these memory locations will contain the band spectrum estimate, and we can then proceed to examine if it is consistent with a flat S."

"Another possible additional test is that we permute the random numbers in some way before Walsh transformation. Given a truly random sequence, we should still get a flat spectrum regardless of what permutation was tried."

1980 -- Atkinson

Atkinson, A. 1980. Tests of Pseudo-random Numbers. *Applied Statistics.* 29(2): 164-171.

"Knuth (1969, Sections 3.3.2 and 3.3.3) divides test of the supposedly independent and uniform U_i into two classes: empirical tests, in which a sample is taken and assessed without consideration of the way in which the pseudo-random numbers are generated, and which do not require a sample. Two theoretical tests are the lattice test (Marsaglia, 1972) and the spectral test described by Knuth (1969, Section 3.3.4)."

"The chief purpose of the present paper is to stress that the lattice test, like the spectral test, is only appropriate for some mixed congruential generators, and to describe how it should be modified for multiplicative generators."

1981 -- Knuth

Knuth, D. 1981. (1st ed. 1969.) *The Art of Computer Programming.* Volume 2: *Seminumerical Algorithms.* Addison-Wesley.

The most-quoted reference, and still a required background, but now somewhat dated.

1983 -- Hopkins

Hopkins, T. 1983. Algorithm AS 193. A Revised Algorithm for the Spectral Test. *Applied Statistics.* 32: 328-335.

"An efficient implementation of an improved algorithm for performing the spectral test (Knuth, 1981) on linear congruential random number generators is presented." "The new algorithm is faster than and does not have the termination difficulties . . . of its predecessor."

1985 -- Marsaglia

Marsaglia, G. 1985. A Current View of Random Number Generators. *Computer Science and Statistics: The Interface.* 3-10. Elsevier Science.

"The ability to generate satisfactory sequences of random numbers is one of the key links between Computer Science and Statistics. Standard methods may no longer be suitable for increasingly sophisticated uses, such as in precision Monte Carlo studies, testing for primes, combinatorics or public encryption number generators: congruential, shift-register and lagged-Fibonacci, give poor results, and describes new methods that pass the stringent tests and seem more suitable for precision Monte Carlo use."

"This theoretical result suggests that if two RNG's produce sequences x_1, x_2, x_3, ... and y_1, y_2, y_3, ... on some finite set S on which we have a binary operation [dot], then the combination generator, producing x_1 [dot] y_1, x_2 [dot] y_2, x_3 [dot] y_3, ... should be better, or at least no worse than, either of the component RNG's." "The binary operation need only have the property that its operation table forms a latin square; it need be neither commutative nor associative."

"A few tests were suggested in the early days of making tables of random digits [5], and M. D. MacLaren and I suggested a few more [8]. These, and a few others, have become a *de facto* standard set of tests, enumerated in Knuth's V2, [6]. Knuth's books are such marvels that they sometimes discourage initiative -- so well done that many readers take them as gospel, the definitive word on the particular subject treated. And so they are, most of the time. But not, I think, for testing random number generators."

1985 -- Beker and Piper

Beker, H. and F. Piper. 1985. *Secure Speech Communications.* Academic Press.

Chapter 3: The Principles of Cryptography, Part IV: Stream Ciphers, Section A: Randomness (p. 104-109). Describes:

1. The Frequency Test.
2. The Serial Test.
3. The Poker Test.
4. The Autocorrelation Test.
5. The Runs Test.

1985 -- Blumer *et. al.*

Blumer, A., *et. al.* 1985. The Smallest Automaton Recognizing the Subwords of a Text. *Theoretical Computer Science.* 40: 31-55. North-Holland.

"In the classic string matching problem for text, we are given a text *w* and a pattern string *x* and we want to know if *x* appears in *w*, i.e., if *x* is a subword of *x*. Standard approaches to this problem involve various methods for preprocessing *x* so that the text *w* can be searched rapidly [1, 9, 16]. Since each search still takes time proportional to the length of *w*, this method is inappropriate when many different patterns are examined against a fixed text . . . . In this case, it is desirable to preprocess the text itself, building an auxiliary data structure that allows one to determine whether *x* is a subword of *w* in time proportional to the length of *x*, not *w*."

1987 -- Rueppel

Rueppel, R. 1987. Linear Complexity and Random Sequences. *Advances in Cryptology -- CRYPTO '86.* 167-188. Springer-Verlag.

"Abstract: The problem of characterizing the randomness of finite sequences arises in cryptographic applications. The idea of randomness clearly reflects the difficulty of predicting the next digit of a sequences from all previous ones. The approach taken in this paper is to measure the (linear) unpredictability of a sequence (finite or periodic) by the length of the shortest linear feedback shift register (LFSR) that is able to generate the given sequence. This length is often referred to in the literature as the *linear complexity* of the sequence. It is shown that the expected linear complexity of a sequence of n independent and uniformly distributed binary random variables is close to n/2 . . . ." "Consequently, we expect a 'typical' random sequence to have associated a 'typical' linear complexity profile closely following the n/2 line."

"*. . . a good random sequence generator should have linear complexity close to the period length, and also a linear complexity profile which follows closely, but 'irregularly', the n/2-line* (where n denotes the number of sequence digits) *thereby exhibiting average step lengths and step heights of 4 and 2, respectively.*"

1987 -- Tezuka

Tezuka, S. 1987. On the Discrepancy of GFSR Pseudorandom Numbers. *Journal of the Association for Computing Machinery.* 34(4): 939-949.

"Abstract. A new summation formula based on the orthogonal property of Walsh functions is devised. Using this formula, the *k*-dimensional discrepancy of the generalized feedback shift register (GFSR) random numbers is derived. The relation between the discrepancy and *k*-distribution of GFSR sequences is also obtained. Finally the definition of optimal GFSR pseudorandom number generators is introduced."

Blumer, A. *et. al.*

Blumer, A. *et. al.* 1987. Complete Inverted Files for Efficient Text Retrieval and Analysis. *Journal of the Association for Computing Machinery.* 34(3): 578-595.

"Abstract. Given a finite set of texts $S = \{w_1, ..., w_k\}$ over some fixed finite alphabet [sigma], a complete inverted file for *S* is an abstract data type that provides the functions *find(w)*, which returns the longest prefix of *w* that occurs (as a subword of a word) in *S; freq(w)*, which returns the number of times *w* occurs in *S;* and *locations(w)*, which returns the set of positions where *w* occurs in *S*."

1988 -- Feldman

Feldman, F. 1987. Fast Spectral Tests for Measuring Nonrandomness and the DES. *Advances in Cryptology -- CRYPTO '87.* 243-254. Springer-Verlag.

"*Abstract* -- Two spectral tests for detecting nonrandomness were proposed in 1977. One test, developed by J. Gait [1], considered properties of power spectra obtained from the discrete Fourier transform of finite binary strings."

"Another test, developed by C. Yuen [2], considered analogous properties for the Walsh transform. In estimating variance of spectral bands, Yuen assumes the spectral components to be independent. Except for the special case of Gaussian random numbers, this assumption introduces a significant error into his estimate."

"A new test, based on an evaluation of the Walsh spectrum, is presented here. This test extends the earlier test of C. Yuen."

"We prove that our measure of the Walsh spectrum is equivalent to a measure of the skirts of the logical autocorrelation function. It is clear that an analogous relationship exists between Fourier periodograms and the circular autocorrelation function."

1988 -- Wanders

Wanders, H. 1988. On the Significance of Golomb's Randomness Postulates in Cryptography. *Philips Journal of Research.* 43(2): 185-222.

"**Abstract.** Golomb's famous book on shift register sequences is a classical reference in the study of stream ciphers. His so-called 'postulates' about PN-sequences are to be generalized and relaxed in real cryptographic applications."

1989 -- Maurer and Massey

Maurer, U. and J. Massey. 1989. Perfect Local Randomness in Pseudo-random Sequences. *Advances in Cryptology -- CRYPTO '89.* 100-112. Springer-Verlag.

"**Abstract.** The concept of provable cryptographic security for pseudo-random number generators that was introduced by Schnorr is investigated and extended. The cryptanalyst is assumed to have infinite computational resources and hence the security of the generators does not rely on any unproved hypothesis about the difficulty of solving a certain problem, but rather relies on the assumption that the number of bits of the generated sequence the enemy can access is limited."

"The results of Section 2 show that provably-secure ciphers can be constructed under the restriction that the number of plaintext bits obtainable by the enemy is smaller than the length of the key, divided by the logarithm of the plaintext length."

"In Section 2, we introduce the concept of a perfect local randomizer, i.e., of a sequence generator that stretches a (binary) random sequence of length *k* to a pseudo-random sequence of length *n* such that every subset of *e* or less bits of the sequence is a set of independent random bits. The concept of a perfect local randomizer corresponds to what is known in combinatorics as an orthogonal array."

1989 -- Richards

Richards, T. 1989. Graphical Representation of Pseudorandom Sequences. *Computers and Graphics.* 13(2): 261-262.

"The technique reviewed in this note provides a simple way of producing two-dimensional graphics from one-dimensional data and also for revealing subtle patterns in noisy data."

"Variations on this technique could be used to examine any sequence of values that appears to be random. If a pattern emerges, the underlying cause may then be investigated."

1989 -- Carroll

Carroll, J. 1989. The binary derivative test: noise filter, crypto aid, and random-number seed selector. *Simulation.* 53(3): 129-135.

"Random noise obscuring digitalized images or text can be removed by a new technique that recognizes the appearance of randomness in short strings. This test makes use of binary derivatives."

"The binary derivative of a string of bits is formed by XORING the members of each successive overlapping pair of bits. One can continue to differentiate the string, losing one bit each time, until only one bit is left."

1989 -- Beth and Dai

Beth, T. and Z-D. Dai. 1989. On the Complexity of Pseudo-Random Sequences -- or: If You Can Describe a Sequence It Can't be Random. *Advances in Cryptology -- EUROCRYPT '89.* 533-543. Springer-Verlag.

"We shall prove in this note that Turing-Kolmogorov-Chaitin complexity and the Linear Complexity are the same for practically all 0-1-sequences of length *n*, already for moderate *n*."

1989 -- Jansen

Jansen, C. 1989. The Shortest Feedback Shift Register That Can Generate A Given Sequence. *Advances in Cryptology -- CRYPTO '89.* 90-99. Springer-Verlag.

"**Abstract.** In this paper the problem of finding the absolutely shortest (possibly nonlinear) feedback shift register, which can generate a given sequence with characters from some arbitrary finite alphabet, is considered. To this end, a new complexity measure is defined, called the maximum order complexity. A new theory of the nonlinear feedback shift register is developed, concerning elementary complexity properties of transposed and reciprocal sequences, and feedback functions of the maximum order feedback shift register equivalent. Moreover, Blumer's algorithm is identified as a powerful tool for determining the maximum order complexity profile of sequences, as well as their period, in linear time and memory. The typical behaviour of the maximum order complexity profile is shown and the consequences for the analysis of given sequences and the synthesis of feedback shift registers are discussed."

1990 -- Feldman

Feldman, F. 1990. A New Spectral Test for Nonrandomness and the DES. *IEEE Transactions on Software Engineering.* 16(3): 261-267.

"*Abstract* -- A new test for detecting the nonrandomness of finite binary strings is proposed. This test, based on an evaluation of the power spectrum of a finite string, extends and quantifies a similar test proposed by Jason Gait [1] in 1977."

"As noted by Gait [1], a good cipher must have the characteristics of a good pseudorandom bit generator. Since most tests of nonrandomness focus on the time domain values of a test string, Gait pointed to the need of also testing frequency domain values. He presented a graphic approach for displaying the power spectrum of binary strings."

"As an empirical measure of the sensitivity of our test, it was compared with a chi-square test for uniformity of distribution, which also measures nonrandomness." "It is also apparent that our spectral tests are sensitive to a different kind of nonrandomness than the chi-square test."

1990 -- Maurer

Maurer, U. 1990. A Universal Statistical Test for Random Bit Generators. *Advances in Cryptology -- CRYPTO '90.* 409-420. Springer-Verlag.

"**Abstract.** A new statistical test for random bit generators is presented that is universal in the sense that any significant deviation of the output statistics from the statistics of a perfect random bit generator is detected with high probability . . . ." "This is in contrast to most presently used statistical tests which can detect only one type of non-randomness . . . ." "Moreover, the new test . . . measures the entropy per output bit of a generator." "The test is easy to implement and very fast."

1990 -- Jansen and Boekee

Jansen, C. and D. Boekee. 1990. On the Significance of the Directed Acyclic Word Graph in Cryptology. *Advances in Cryptology: AUSCRYPT '90.* 318-326. Springer-Verlag.

"**Summary.** Blumer's algorithm can be used to build a Directed Acyclic Word Graph (DAWG) in linear time and memory from a given sequence of characters. In this paper we introduce the DAWG and show that Blumer's algorithm can be used very effectively to determine the maximum order (or nonlinear) complexity profile of a given sequence." "It also appears that the DAWG is an even more efficient means of generating the sequence, given a number of characters, then e.g. the nonlinear feedback shift register equivalent of the sequence, as it always needs the least amount of characters to generate the remainder of the sequence."

"In [Jans 89] a new complexity measure for sequences is proposed, called Maximal Order Complexity, which is equal to the length of the shortest (possibly nonlinear) feedback shift register that can generate a given sequence. Analogous to Rueppel's linear complexity profile [Ruep 84], the maximum order complexity profile is proposed as a measure of 'goodness' for sequences, i.e. a measure which shows how well a given sequence resembles a real random sequence."

1990 -- Jansen and Boekee

Jansen, C. and D. Boekee. 1990. A Binary Sequence Generator Based on Ziv-Lempel Source Coding. *Advances in Cryptology -- AUSCRYPT '90.* 156-164. Springer-Verlag.

"**Summary.** A new binary sequence generator is proposed, which is based on Ziv-Lempel source coding. In particular, the Ziv-Lempel decoding algorithm is applied to codewords generated by linear feedback shift registers. It is shown that the sequences generated in this way have high linear complexity and good statistical properties."

1991 -- Compagner

Compagner, A. 1991. The Hierarchy of Correlations in Random Binary Sequences. *Journal of Statistical Physics.* 63(5/6): 883-896.

"The meaning of randomness is studied for the simple case of binary sequences. Ensemble theory is used, together with correlation coefficients of any order. Conservation laws for the total amount of correlation are obtained. They imply that true randomness is an ensemble property and can never be achieved in a single sequence."

1991 -- Mund

Mund, S. 1991. Ziv-Lempel Complexity for Periodic Sequences and its Cryptographic Application. *Advances in Cryptology -- EUROCRYPT '91.* 114-126. Springer-Verlag.

"In the last couple of years several different complexity measures were used to examine pseudorandom number sequences in cryptography. Examples for such complexity measures are the linear complexity which is defined in Rueppel [Ruep86] or the maximal-order complexity which was introduced by Jansen [Jans89]." "Another complexity measure for sequences was defined by Ziv and Lempel in 1976 [Lemp76]. This complexity measure is a measure of the rate at which new patterns emerge as we move along the sequence."

1991 -- Compagner

Compagner, A. 1991. Definitions of Randomness. *American Journal of Physics.* 59(8): 700-705.

"All numbers are arbitrary, but some are more arbitrary than others. Below one million, 170769 appears to be more random than 888888, though *a priori* they are equally probable."

"When randomness is not understood, it is more difficult to achieve than law and order." ". . . the subject suffers from an excess of mathematical ingenuity which has made straightforward ideas on randomness obsolete before they ever were formulated properly."

"[The new] ideas are based on ensemble theory and give rise to new definitions." ". . . when ensembles are used for the foundations of probability theory, randomness has to be identified with uncorrelatedness, a neglected notion that yet solves many puzzles surrounding randomness."

1992 -- L'Ecuyer

L'Ecuyer, P. 1992. Testing Random Number Generators. *Proceedings of the 1992 Winter Simulation Conference.* 305-313.

"**ABSTRACT.** So-called Random number generators on computers are deterministic functions producing a sequence of numbers which should *mimic* a sample of i.i.d. [individually independently distributed] *U*(0,1) random variables. Two classes of tests are commonly applied to such generators. Firstly, the theoretical tests, which look at the intrinsic structure of the generator to derive behavioral properties of the sequence of points . . . ." "Secondly, the empirical goodness-of-fit tests, which try finding statistical evidence against the null hypothesis: 'the sequence is a sample from i.i.d. *U*(0,1) random variables'. In this paper, we survey the main techniques from both classes, discuss their philosophy, and look at some of the most recent developments . . . ."

1993 -- Marsaglia and Zaman

Marsaglia, G. and A. Zaman. 1993. Monkey Tests for Random Number Generators. *Computers and Mathematics with Applications.* 26(9): 1-10.

"Few images invoke the mysteries and ultimate uncertainties of a sequence of random events as well as that of the proverbial monkey at a typewriter." "Technically, we are concerned with overlapping *m*-tuples of successive elements in a random sequence."

"This article describes some very simple, as well as some quite sophisticated, tests that shed light on the suitability of certain random number generators."

1993 -- Maclaren

Maclaren, N. 1993. Cryptographic Pseudo-random Numbers in Simulation. *Fast Software Encryption.* Ross Anderson, ed., 185-190.

"A fruitful source of confusion on the Internet is that both cryptologists and statisticians use pseudo-random numbers, but that their objectives and constraints are subtly different."

"It is important to note that there is no consensus on when a pseudo-random number generator can be regarded as adequate, both because the theory is very incomplete and because so many different fields are involved."

1995 -- Gustafson, Dawson and Golic

Gustafson, H, E. Dawson, and J. Golic. 1995. Randomness Measures Related to Subset Occurrence. *Cryptography: Policy and Algorithms.* 132-143. Springer-Verlag.

"**Abstract** Statistical tests have been applied to measures obtained from partitioning the keystream of a stream cipher into subsets of a given length. Similarly, the strength of a block cipher has been measured by applying statistical tests to subsets obtained from both the input and output blocks. There are problems in applying these tests as the size of the subsets increases. We propose a novel method based on the classical occupancy problem to deal with larger subsets in testing for randomness in a keystream in the case of a stream cipher and for independence between subsets of input and output blocks in the case of a block cipher."

Terry Ritter, his current address, and his top page.

Last updated: 1996-08-15

Latin Squares: A Literature Survey

Research Comments from Ciphers By Ritter

Terry Ritter

A **Latin square** of order *n* is an *n* by *n* array containing symbols from some alphabet of size *n*, arranged so that each symbol appears exactly once in each row and exactly once in each column. The best introduction here is in Bose and Manvel.

It seems that Latin squares were originally mathematical curiousities, but serious statistical applications were found early in the 20th century, as "experimental designs." The classic example is the use of a Latin square configuration to place 3 or 4 different grain varieties in test patches. Having multiple patches for each variety helps to minimize localized soil effects. Similar statements can be made about medical "treatments."

In cryptographic application, a Latin square can be seen as a stream cipher *combiner*, a sort of generalized exclusive-OR function. A Latin square combiner is "invertible" provided one of the inputs to the combiner is known, as is generally the case for a cryptographic combiner. Advantages of the Latin square include a massive internal state which may be keyed, and the fact that combining operations using a keyed square are nonlinear. Architectures which have little worth with linear combining, such as a sequence of combinings, or a selection between combinings, are new opportunities when we have nonlinear combiners with substantial internal state.

Contents

- 1934
  - Fisher and Yates, The 6 x 6 Squares. The article enumerates the squares, but we extract only a few comments here.
- 1939
  - Norton, The 7 x 7 Squares. Big. Perhaps the last attempt to enumerate squares in print. Here we extract essentially a glossary of terms.
  - Bose, On the Construction of Balanced Incomplete Block Designs. A big, famous article, but too complex to extract very much.
  - Stevens, The Completely Orthogonalized Latin Square.
- 1942
  - Mann, The Construction of Orthogonal Latin Squares. These turn out to be important in Ritter's Balanced Block Mixing designs.
- 1949
  - Shannon, Communication Theory of Secrecy Systems. Shannon defines perfect security and identifies that with a Latin square.
- 1952
  - Bush, Orthogonal Arrays of Index Unity. Here we have an introduction to orthogonal arrays.
  - Bose and Bush, Orthogonal Arrays of Strength Two and Three. A few extracts.
- 1960
  - Bose, Chakravarti and Knuth, On Methods of Constructing Sets of Mutually Orthogonal Latin Squares Using a Computer. II. Apparently somebody was interested in finding orthogonal Latin squares; wonder why?
- 1961
  - Johnson, Dulmage and Mendelsohn, Orthomorphisms of Groups and Orthogonal Latin Squares. An approach to the sub-structure of a Ls order.
  - Yamamoto, Generation Principles of Latin Squares. More delving into sub-structure.
- 1962
  - Collison, Magic Square algorithms in Algol, but no extracts here.
- 1963
  - O'Carroll, A Method of Generating Randomized Latin Squares. The full description is here, but backtracking will be required to get to order 256.
- 1967
  - Wells, The Number of Latin Squares of Order 8. The order is too big to enumerate in print. Here we have extracts on how a count is done.
- 1971
  - Wells, the book. Very lofty, however.
- 1975
  - Bammel and Rothstein, The Number of 9 x 9 Latin Squares. More extracts on how a count is done, and a few timing values.
  - Alter, How Many Latin Squares Are There? Oddly, still an open question in mathematics. Here we have counts of reduced squares for each order, with the corresponding prime factorization.
- 1980
  - Encyclopedic Dictionary of Mathematics Links Latin squares to triplets and quasigroups, and gives counts for both reduced and nonisomorphic squares.
- 1984
  - Bose and Manvel, Introduction to Combinatorial Theory. Here we have an introduction to Latin squares.
- 1987
  - Stein, Large Sample Properties of Simulations Using Latin Hypercube Sampling.
  - Kull and Specker, Direct Construction of Mutually Orthogonal Latin Squares. The algorithms here are in math and set notation, and are much too general and detailed to get right if included here.
  - Massey, Maurer and Wang, Non-Expanding, Key-Minimal, Robustly-Perfect, Linear and Bilinear Ciphers. A contemporary cryptographic use of Latin squares.
  - Hunter, Are Some Latin Squares Better Than Others? Mainly a statistics issue, but the article has an interesting graph of order-3 sub-structure.
- 1989
  - Wolf, Nondeterministic Circuits, Space Complexity and Quasigroups. Included here for a view of quasigroups.
- 1990
  - Godsil and McKay, Asymptotic Enumeration of Latin Rectangles.
- 1991
  - Byers, Basic Algorithms for Random Sampling and Treatment Randomization. While not particularly impressive programming, the basic idea of shuffling the rows and columns of an easy-to-produce square can be very useful.
  - Denes and Keedwell, Latin Squares: New Developments in the Theory and Applications. Perhaps the major modern book, covering a range of uses and related ideas.
  - Camion, Carlet, Charpin and Sendrier, On Correlation-Immune Functions. Identifies orthogonal arrays as correlation-immune combining functions.
- 1992
  - Shao and Wei, A formula for the number of Latin squares.
- 1995
  - McKay and Rogoyski, Latin Squares of Order 10.
- 1996
  - Jacobson and Matthews, Generating uniformly distributed random latin squares.

1934 -- Fisher and Yates

Fisher, R. and F. Yates. 1934. The 6 x 6 Latin Squares. *Proceedings of the Cambridge Philosophical Society.* 30: 492-507.

"The problem of the enumeration of the different arrangements of *n* letters in an *n* x *n* Latin square, that is, in a square in which each letter appears once in every row and once in every column, was first discussed by Euler(1)."

"The problem of the Latin square has become of practical interest in recent years in conjunction with the development of an adequate theoretical basis for the design of biological experiments . . . ." "The reason for its special suitability lies in its satisfactorily fulfilling two distinct requirements: (1) in equalising . . . the fertility of the land on which the different treatments are to be tested, and (2) in allowing . . . of a random choice among the different possible squares which could be laid down on the same area. This element of randomisation is now recognized to be a necessary condition for the validity of the estimate of error by which the results of the experiment are to be judged . . . ."

1939 -- Norton

Norton, H. 1939. The 7 x 7 Squares. *Annals of Eugenics.* 9: 269-307.

"Latin squares were first studied by Euler near the end of the eighteenth century, and have since been investigated rather widely but with small success."

Latin square

"A Latin square of side *n* is a square arrangement of n^2 letters, *n* of each of *n* kinds, such that each letter occurs once in each row and once in each column."

Cyclic Latin square

"An *n* x *n* Latin square in which each row is derived from any other in a cyclic permutation of degree *n*, or by a power of such a permutation, is a *cyclic* Latin square."

Standardization

"If the letters of the top row and of the left column of a Latin square are in some standard order, alphabetic order being convenient, the square is said to be *standard* or *in standard form*. It follows that *n*!(*n*-1)! Latin squares of side *n* may be represented by a single standard square."

Constraints

"Since the elements of a Latin square are constrained to certain rows and columns and letters, rows, columns and letters will be called *constraints*, and a Latin square may be said to be a square of three constraints."

Orthogonal Latin squares

"It may be possible to find two Latin squares of the same size, necessarily not both standard, such that when one is superposed on the other, each letter of the one coincides once with each letter of the other. Such two Latin squares are said to be *orthogonal*." "If one of the two Latin squares be written in Greek letters and superposed on the first, the resulting square is called a *Graeco-Latin square*, and may be said to be a square of four constraints. It may be possible to go further, but a group of mutually orthogonal squares of side *n* can number *n* - 1 at most."

Orthogonal Standardization

"The requirements of the Graeco-Latin square do not permit that both of the component Latin squares be standard. The standard form of a Graeco-Latin and higher squares is therefore defined to be that in which the basic Latin square is standard and the superposed square or squares have the top row in standard order."

Transformations

"Any permutation of the rows, columns or letters of a Latin square, or any combination of such permutations, is called a *transformation*."

Conjugacy

"Two Latin squares are said to be *conjugate* if the rows of one are the columns of the other. In other words, if the rows and columns of a square be interchanged, the conjugate square is generated."

Adjugacy

"Adjugacy is a generalization of the concept of conjugacy and is used here to include conjugacy. Two squares are said to be *adjugate* if a permutation of the constraints of one generates another."

Species

"A *species* of Latin squares is a group of adjugate sets such that any permutation of the constraints of any member generates a member, and all possible permutations of the constraints of any member generates all members."

Intercalate

"An intercalate is a Latin square of side 2 embedded in a larger square. For example, the following 7 x 7 Latin square has 10 intercalates, one of which contains the elements 21B, 24G, 71G and 74B."

```
A B C D E F G
B E D G F C A
C A B E D G F
D G F C B A E
E C G F A D B
F D E A G B C
G F A B C E D
```

Intercalate reversal

"It is obvious that the two letters, or the two rows or the two columns, involved in an intercalate may be interchanged within the intercalate but not elsewhere, or *visa versa*, while preserving the Latin square." "This interchange is called *intercalate reversal*, and usually generates a member of a new species."

Groupings of species

"A *family* of species is a group such that the reversal of any intercalate contained in the group generates a member of the group, and the reversal of all intercalates contained in the group generates all members of the group. A *domain* is exactly analogous to a family, but includes reversal of generalized intercalates. The *universe* of squares of side *n* contains all the squares of that size and of a specified number of constraints."

"The enumeration of Latin squares was first undertaken by Euler (1782) in searching for a 6 x 6 Graeco-Latin square. He found 1, 1, 4, and 56 standard Latin squares of sides 2, 3, 4 and 5 respectively, using an exhaustive process."

"Fisher & Yates (1934) enumerated the 6 x 6 Latin squares by a method different from that of Tarry, and found 9408 squares of 17 types, none of the 17 types having Graeco-Latin solutions." "They point out that there are 22 sets of which 7 are self-adjugate and the remainder belong to 5 adjugate triplets. Each triplet contains a conjugate pair, and if their identity is recognized but not that of the adjugate member, there appear to be 17 types, whereas recognizing adjugacy there are seen to be only 12 species."

1939 -- Bose

Bose, R. 1939. On The Construction of Balanced Incomplete Block Designs. *Annals of Eugenics.* 9: 353-399.

"The interest of mathematicians in combinatorial problems, involving the arrangement of a finite number of things in sets or patterns, satisfying given conditions, can be traced back to at least as far as Euler (1782), who interested himself in the construction of Latin and Graeco-Latin squares. Steiner (1853) proposed the problem of arranging *N* things in triplets, such that every pair occurs in just one and only one triplet. Such an arrangement may be called a *simple triplet system* or a *Steiner's triplet system*." "The object of this paper is to study the combinatorial problem involved in the construction of a certain type of design, first introduced in experimental studies by F. Yates (1936), and called *Balanced Incomplete Block Design*. An example of such a design is afforded by a simple triple system."

"If *v* varieties or treatments are to be compared in randomized blocks of *k* experimental units (*k*<*v*), then block differences can be simply eliminated, if the arrangement is such that every two varieties occur together in the same number [lambda] of blocks. It is readily seen that the integers *v*, *b*, *r*, *k*, [lambda] satisfy the equations

bk = *vr*, [lambda](*v*-1) = *r*(*k*-1),

so that these equations provide necessary conditions, for the existence of a balanced incomplete block design, with *v* varieties arranged in *b* blocks of *k* units each, each variety being replicated *r* times, and each pair occurring [lambda] times. These equations do not, however, provide sufficient conditions . . . ."

1939 -- Stevens

Stevens, W. 1939. The Completely Orthogonalized Latin Square. *Annals of Eugenics.* 9: 82-93.

The Latin square

"A Latin square of side *s* is an arrangement of *s* letters, *s* of each *s* kinds, such that in each row and in each column each letter occurs exactly once. Thus when *s*=3, a suitable arrangement is"

```
A  B  C
B  C  A
C  A  B
```

The Graeco-Latin Square

"It may or may not be possible to write down another Latin square with the same letters such that when the two squares are superimposed, each letter of one square coincides exactly once with each letter of the other square. Two squares of side 3 with this property are"

```
A  B  C      A  B  C
B  C  A      C  A  B
C  A  B      B  C  A
```

"When the second square is written in Greek letters and superimposed on the first, the composite is called a Graeco-Latin square. In general, any two such Latin squares are said to be orthogonal to each other."

Mutually orthogonal Latin squares

"The number of squares in a set of orthogonal squares of side *s* is not greater than (*s*-1). For the letters of each square may be permuted among themselves without destroying the property of orthogonality. Let this be done so that the top line of each square commences *A B* .... Then, since *B* coincides with itself in the top row, it follows that in the first column *B* cannot occupy any position twice or the top position at all. Hence there are not more than (*s*-1) positions available for *B*."

"The same fact is obvious from consideration of Analysis of Variance. For differences between the "plots" with the same letter in one Latin square, or between rows or columns account for (*s*-1) degrees of freedom, and since there are altogether s^2-1 degrees of freedom, it follows that there are not more than *s*+1 independent methods of subdivision. Of these rows and columns account for two, leaving only *s*-1 different subdivisions by letters."

Construction of a complete set of orthogonal squares

"Assuming that a field of *s* letters exist, we may write a square of letters as

$\{u_{[lambda]}u_x+u_y\}$

$u_{[lambda]}$ = constant <> 0

$u_x,u_y = 0,1,u_2,...u_{s-1}$

where $u_{[lambda]}u_x+u_y$ is the letter in row u_x and column u_y."

1942 -- Mann

Mann, H. 1942. The Construction of Orthogonal Latin Squares. *Annals of Mathematical Statistics.* 13: 418-423.

"A Latin square is an arrangement of *m* variables $x_1, x_2, ..., x_m$ into *m* rows and *m* columns such that no row and no column contains any of the variables twice. Two Latin squares are called orthogonal if when one is superimposed upon the other every ordered pair of variables occurs once in the resulting square."

Definition 1

"*If A is orthogonal to B, and if in the reduced form the permutations of A are the same as B in a different order, and if these permutations form a group G, then the pair A and B is said to be based on the group G.*"

"Most of the sets of orthogonal Latin squares that have been constructed so far are based on abelian groups of the type (*p*,*p*,...,*p*) and the mappings of the squares of the sets into each other are automorphisms of this group."

"Below are given two examples of Graeco-Latin squares obtained from complete mappings which are not obtained from automorphisms. Neither could have been obtained by combining Graeco-Latin squares constructed by the method of Bose [1] and Stevens [2]."

1949 -- Shannon

Shannon, C. 1949. Communication Theory of Secrecy Systems. *Bell System Technical Journal.* 28: 656-715.

"Perfect systems in which the number of cryptograms, the number of messages, and the number of keys are all equal are characterized by the properties that (1) each *M* is connected to each *E* by exactly one line, (2) all keys are equally likely. Thus the matrix representation of the system is a "Latin square." (p. 681)

1952 -- Bush

Bush, K. 1952. Orthogonal Arrays of Index Unity. *Annals of Mathematical Statistics.* 23: 426-434.

"In this paper we shall proceed to generalize the notion of a set of orthogonal Latin squares, and we term this extension an orthogonal array of index unity. In Section 2 we secure bounds for the number of constraints which are the counterpart of the familiar theorem which states that the number of mutually orthogonal Latin squares of side *s* is bounded above by *s* - 1. Curiously, our bound depends upon whether *s* is odd or even. In Section 3 we give a method of constructing these arrays by considering a class of polynomials with coefficients in the finite Galois field *GF*(*s*), where s is a prime or a power of a prime."

Let a set of *s* integers, 0,1,...,*s*-1 be arranged in an *s* x *s* square in such a way that every integer occurs *s* times. If each integer occurs once and only once in every row and column, the square is said to be a Latin square of side *s*. Two squares are said to be orthogonal to one another if, when one square is superimposed upon the other square, every number of the first occurs once and only once with every number of the second square. The set of at most *s* - 1 Latin squares which are mutually orthogonal, we may adjoin two other squares which are not Latin squares, but which are orthogonal to each other and to every other Latin square in the orthogonal set. The first of these squares is constructed by taking each element of the first row as 0, each element of the second row as 1, and so on. The second square is the transpose of the first square. Thus a total of *s* + 1 orthogonal squares is possible at best, and it is known that this bound is attainable when *s* is a prime or a power of a prime [1]. When this bound is attained, we say that we have a complete set of orthogonal squares. As an example of a complete set, we might choose *s* = 3 and write

```
0 0 0   0 1 2   0 1 2   0 1 2
1 1 1   0 1 2   1 2 0   2 0 1
2 2 2   0 1 2   2 0 1   1 2 0
```

"If we write in order the elements of each square in a line, we can display these squares in the following form:"

```
0 0 0 1 1 1 2 2 2      [first square]
0 1 2 0 1 2 0 1 2      [second square]
0 1 2 1 2 0 2 0 1      [third square]
0 1 2 2 0 1 1 2 0      [fourth square]
```

"In this form we see that any two rows have the property that each one of the nine possible ordered pairs occurs exactly once when one row is superimposed on another row. We now generalize this basic idea."

"Let us consider a matrix *A* = [a_{ij}], where each a_{ij} represents one of the integers 0,1,...,*s*-1, *s* > 1. The matrix is rectangular with *N* columns, which we shall call the blocks of the array, and *k* rows. Consider all *t*-rowed submatrices of *N* columns which can be formed from this array, *t* <= *k*. Each column of any *t*-rowed submatrix can be regarded as an ordered *t*-plet, so that each *t*-rowed submatrix contains *N* such *t*-plets. The matrix *A* will be called an orthogonal array [*N*,*k*,*s*,*t*] of size *N*, *k constraints, s levels, strength t and index [lambda]* if each of the C^k_t *t*-rowed *N*-columned submatrices that may be formed from the array contains every one of the s^t possible ordered *t*-plets each repeated [lambda] times. It is clear that this definition implies that each row contains the *s* integers 0,1,...,*s*-1, each repeated [lambda]s^{t-1} times. We shall consider the case where [lambda] = 1 and refer to such arrays as "orthogonal arrays of index unity."

1952 -- Bose and Bush

Bose, R. and K. Bush. 1952. Orthogonal Arrays of Strength Two and Three. *Annals of Mathematical Statistics.* 23: 508-524.

"Orthogonal arrays can be regarded as natural generalizations of orthogonal Latin squares . . . ."

"A *k* x *N* matrix *A*, with entries from a set [sigma] of *s* >= 2 elements, is called an orthogonal array of strength *t*, size *N*, *k* constraints and *s* levels if each *t* x *N* submatrix of *A* contains all possible *t* x 1 column vectors with the same frequency [lambda]. The array may be denoted by (*N*,*k*,*s*,*t*). The number [lambda] may be called the index of the array. Clearly *N* = [lambda]s^t."

1960 -- Bose, Chakravarti and Knuth

Bose, R, I. Chakravarti and D. Knuth. 1960. On Methods of Constructing Sets of Mutually Orthogonal Latin Squares Using a Computer. II *Technometrics.* 3(1): 111-117.

"In [1] a method of searching for sets of mutually orthogonal Latin squares of size *v* = 4*t* where 4*t* is the order of a Hadamard matrix is given. The method is based on the concept of orthogonal mappings of a group, due to Mann [4].

1961 -- Johnson, Dulmage and Mendelsohn

Johnson, D., A. Dulmage and N. Mendelsohn. 1961. Orthomorphisms of Groups and Orthogonal Latin Squares. I *Canadian Journal of Mathematics.* 13: 356-372.

"It is a trivial fact that for any *n*, there are at most *n* - 1 mutually orthogonal latin squares. When *n* - 1 such squares exist we say that the set of squares is complete. There is an easily established 1-1 correspondence between complete sets of orthogonal latin squares and finite affine (and hence projective) plane geometries." "The basic square is the group addition table of an elementary abelian group and the remainder of the squares are obtained by a set of permutations of the rows in each of which the first row is kept fixed."

1961 -- Yamamoto

Yamamoto, K. 1961. Generation Principles of Latin Squares. *Bulletin of the International Statistical Institute.* 38(4): 73-76.

"In the universe *L* of Latin squares of order *n*, the transformation group *G*, comprising all permutations of rows, columns and treatments, and all the six adjugations, is usually utilized to subdivide *L* into finer classes. But the group *G* may also be regarded as a generation principle to derive a new Latin square from a given one."

1962 -- Collison

Collison, D. 1962. Algorithm 117. Magic Square (Even Order). Algorithm 118. Magic Square (Odd Order). *Communications of the ACM.* 5(8): 435, 436.

Algorithms in Algol.

1963 -- O'Carroll

O'Carroll, F. 1963. A Method of Generating Randomized Latin Squares. *Biometrics.* December. 652-653.

"The square is constructed row by row, but within each row the elements are not necessarily entered in regular order. Consider the position when, in constructing an *N* x *N* Latin square, the first *R* - 1 rows have been completed and in the *R*th row *I* - 1 elements have been filled in. Let A_J be the number of different letters that can be inserted in the *J*th position in the *R*th row without violating the condition for a Latin square. If the *J*th position has already been filled then A_J = 0, otherwise *J* is obtained by counting the number of different letters that have not already been included in either the *R*th row or the *J*th column. Let A_{N+K} be the number of different positions in the *R*th row in which the *K*th letter of the alphabet can be inserted. If the *K*th letter of the alphabet has already been used in this row then A_{N+K} = 0. Let A_S be the smallest non-zero value in the set $A_1,A_2,...,A_{2N}$, and *B* be a random integer in the range 1 to A_S. (If there is a tie, the choice of A_S among the subset of jointly smallest elements is immaterial; that with the smallest value of *S* within this subset can conveniently be taken). The next step in constructing the square is then determined by the values of *S* and *B*, as follows"

"(i) If *S* <= *N*, insert in the *S*th position in the *R*th row the *B*th letter among those that can be entered in this position."

"(ii) If *S* > *N*, insert the (*S* - *N*)th letter of the alphabet in the *B*th position among those still open to it in the *R*th row."

1967 -- Wells

Wells, B. 1967. The Number of Latin Squares of Order 8, *Journal of Combinatorial Theory.* 3: 98-99.

"In 1934, Fisher and Yates [1] enumerated Latin squares of order six, giving 9408 as the number of reduced squares. (A Latin square is *reduced* when the first row and first column are in lexicographic order.) They arrived at that total as a by-product in the explicit construction of equivalence class representatives; there are, in fact, 12 "species" of 6 x 6 Latin squares. A *species* is a class of squares equivalent under arbitrary row, column, or label permutation ($(n!)^3$ of them) or permutation of the three "dimensions," the rows, the columns, and the letters (3! of them -- giving $6(n!)^3$ transformations in all).

"In 1948, Sade [3] enumerated reduced 7 x 7 squares by a method which by-passes the construction of species representatives. (His total of 16,942,080, however, did lead to the discovery of the 147th species [4] which had been overlooked by Norton [2].) Sade's method is to successively calculate for *k* = 1,2,...,*K*, *K* <= *n*, a (complete) set of *reduced k* x *n* Latin rectangles (the first column is not only lexicographic but contains the labels 1,2,...,*k*) inequivalent under row, column or label permutation, keeping track of the number of ways the rectangle could have been formed. The (*k* + 1)-row rectangles are formed by adding a row to each *k*-row rectangle in all possible ways, eliminating the equivalent rectangles (actually the difficult part of the calculation) as they appear. It is not necessary, or efficient, to continue the process until *k* = *n*. When *k* = *K* (Sade used *K* = 4), one may sum the product of the number of ways in which a rectangle could have been formed (already known) and the number of ways the rectangle could have been completed to a square (easily computed) over the inequivalent *k*-row rectangles, producing the number of *n* x *n* squares.

"Using a computer adaptation of Sade's method, the author has enumerated 8 x 8 reduced Latin squares, I_8, finding I_8 = 535,281,401,856."

1971 -- Wells

Wells, M. 1971. *Elements of Combinatorial Computing.* Ch. 7, 198-206. Pergamon Press.

A *very* esoteric presentation of algorithms in math and set notation (as opposed to a computing language).

"The enumeration of reduced Latin squares is an excellent example of the application of the equivalence algorithm just discussed (and of the branch merging technique of section 4.4.4)." (p. 203)

1975 -- Bammel and Rothstein

Bammel, S. and J. Rothstein. 1975. The Number of 9 x 9 Latin Squares. *Discrete Mathematics.* 11: 93-95.

"Sade [1] enumerated the reduced 7 x 7 Latin squares, and by adapting his method to the Maniac II computer, Wells [2,3] enumerated the reduced 8 x 8 squares. We used an improved algorithm for determining equivalence classes in Latin rectangles, which made enumeration of the 9 x 9 squares feasible."

"Sade's method is based on the fact that equivalent Latin rectangles can be filled out to complete n x n Latin squares in the same number of ways (have the same count) where equivalent rectangles are intertransformable by column and label permutations such that corresponding columns have the same set of (unordered) symbols. The number of squares is then the sum of counts over all equivalence classes. The count of a *k* x *n* rectangle is computed by forming all (*k*+1) x *n* rectangles from it, dividing them into equivalence classes, repeating the process for all (*k*+1) x *n* rectangles for one representative of each class, and so on, until a convenient stage (usually *k* = *n*-3 or *n*-4) is reached for exhaustive enumeration. Then knowing the counts for (*k*+1)-row classes, the counts for *k*-row classes are easily computed. The total number of reduced squares is given by *k* = 1.

"A modest improvement in algorithm efficiency is achieved by increasing equivalence class size. We note that if each row of a rectangle be regarded as a permutation of the top row, then the rectangle obtained by replacing each permutation by its inverse has the same count. Following Wells, we work with incidence matrices but now our rectangles are in the same class if their incidence matrices are equivalent under transposition as well as row and column permutation.

"A major improvement in efficiency is achieved in the incidence matrix equivalence algorithm by using a lexicographic procedure which eliminates indeterminacy and back-tracking, normally a time-consuming process. The rows and columns of the incidence matrix are sorted and partitioned according to Wells' row and column weights, but now if the set of column weights is lexicographically less than the set of row weights, the matrix is transposed. At each step sub-matrices (larger than 1 x 1) resulting from previous partitioning are individually sorted and partitioned in a similar manner where the resulting sorting and partitioning is applied to the entire matrix. It frequently happens that "deadlock" occurs in that there exist sub-matrices larger than 1 x 1 within which rows and columns cannot be differentiated. In this case, we arbitrarily break deadlock by partitioning one of the sub-matrices between its first and second columns and then proceed normally. At each step, the sub-matrix to be operated on is selected by a set of well-defined rules. The matrix is in "conventional form" when it is completely partitioned.

"Note that each matrix is reduced to conventional form only once before searching the table of previously encountered conventional forms (representing equivalence classes). This affords another major improvement in that it is not necessary to attempt to reduce one matrix to another at each comparison."

Table 1

n	reduced Latin squares	computation time
7	16 942 080	30 sec.
8	535 281 401 856	4 min.
9	377 597 570 964 258 816	4.7 hours

1975 -- Alter

Alter, R. 1975. How Many Latin Squares Are There? *American Mathematical Monthly.* 82: 632-634.

"A latin square of order *n* is an arrangement of the first *n* integers in an *n* x *n* array so that every integer appears exactly once in every row and exactly once in every Column. Let L_n be the number of latin squares of order *n* and let R_n be the number of reduced latin squares of order *n*. (A reduced latin square has the first row and first column in natural, i.e., lexicographic, order). It is easy to see that

(1) $L_n = n!(n-1)!R_n$.

"To answer the title question it suffices to determine R_n."

Table 1

n	R_n	Prime Factorization
1	1	1
2	1	1
3	1	1
4	4	2^2
5	56	$2^3 . 7$
6	9 408	$2^6 . 3 . 7^2$
7	16 942 080	$2^{10} . 3 . 5. 1103$
8	535 281 401 856	$2^{17} . 3 . 1 361 291$
9	377 597 570 964 258 816	$2^{21} . 3^2 . 5 231 . 3 824 477$

"With current computing facilities, Table 1 will undoubtably be extended. This in turn may shed some light on the above divisibility questions. However, the determination of R_n (and thus also L_n) still appears to be extremely difficult."

1980 -- EDM

Combinatorial Theory. *Encyclopedic Dictionary of Mathematics.* Mathematical Society of Japan. MIT Press. 1980.

"Let [omega] = {$a_1,a_2,...,a_n$} be a set of *n* symbols. An *n* x *n* square array *L* of n^2 symbols taken out of [omega] is called a **Latin square** over [omega] (or Latin square of **order** *n*) if there is no repetition of symbols in each row and in each column of *L*. It is convenient to identify the row and column index sets with [omega]. Thus, we may write *z*=*x.y* if the entry of *L* in the row *x* and column *y* is the symbol *z*. Then we can describe the Latin square as a binary operation system ([omega],.) . . . ." "On the other hand, if we take the n^2 triplets *xyz* formed in this way, then the set of all these triplets forms an error-correcting code . . . ." A Latin square *L* over [omega] is called **reduced** (or in standard form) if both the row a_1 and the column a_1 consist of the natural sequence $a_1,a_2,...,a_n$. Thus the binary operation system ([omega],.) has an identity element a_1, and hence is a quasigroup. The number of Latin squares over [omega] is *n*!(*n*-1)! times the number of *L*(*n*) of reduced Latin squares of order *n*. The number *L*(*n*) has been calculated for *n* <= 8; *L*(1) = *L*(2) = *L*(3) = 1, *L*(4) = 4, *L*(5) = 56, *L*(6) = 9,408, *L**(7) = 16,942,080, *L*(8) = 435,281,401,856, *L*(9) = 377,597,570,964,258,816. Two Latin squares over the same set [omega] are called **isomorphic** if one is obtained from the other by a combination of permutations of rows, columns, and the entry alphabets. The number *L**(*n*) of nonisomorphic Latin square of order *n* has been calculated for *n* <= 8; *L**(1) = *L**(2) = *L**(3) = 1, *L**(4) = *L**(5) = 2, *L**(6) = 22, *L**(7) = 563, *L**(8) = 1,676,257." (p. 233)

1984 -- Bose and Manvel

Bose, R. and B. Manvel. 1984. *Introduction to Combinatorial Theory.* 135-149 (Ch. 7). John Wiley & Sons.

"A *Latin square* of order *s* is identified as an *s* x *s* square, the s^2 cells of which are occupied by *s* distinct symbols . . . such that each symbol occurs once in each row and once in each column."

"Two Latin squares of the same order are said to be *orthogonal* if, on superposition, each symbol of the first square occurs exactly once with each symbol of the second square."

"A set of Latin squares (all of the same order), any two of which are orthogonal, is said to be *a set of mutually orthogonal Latin squares.*

"A Latin square is said to be in the *standard form* if the symbols in the initial row are in the natural order.

"A Latin square can always be brought to the standard form by renaming the symbols. If two Latin squares are orthogonal, the renaming can be done independently for each square without destroying orthogonality."

"There are exactly two Latin squares of order 2."

```
0 1   1 0
1 0   0 1
```

"Next consider Latin squares of order 3. We shall show that once the initial row and column are filled, there is one way to complete the Latin square."

```
0 1 2
1 2 0
2 0 1
```

"We can permute the columns in six different ways. Corresponding to each of these permutations there are two ways to permute the rows other than the initial row. Hence there are twelve distinct Latin squares of order 3."

```
0 1 2   0 2 1   1 2 0   1 0 2   2 0 1   2 1 0
1 2 0   1 0 2   2 0 1   2 1 0   0 1 2   0 2 1
2 0 1   2 1 0   0 1 2   0 2 1   1 2 0   1 0 2

0 1 2   0 2 1   1 2 0   1 0 2   2 0 1   2 1 0
2 0 1   2 1 0   0 1 2   0 2 1   1 2 0   1 0 2
1 2 0   1 0 2   2 0 1   2 1 0   0 1 2   0 2 1
```

"There are exactly four Latin squares of order 4 in which the initial row and the initial column are in the natural order . . . ."

```
0 1 2 3   0 1 2 3   0 1 2 3   0 1 2 3
1 2 3 0   1 3 0 2   1 0 3 2   1 0 3 2
2 3 0 1   2 0 3 1   2 3 0 1   2 3 1 0
3 0 1 2   3 2 1 0   3 2 1 0   3 2 0 1
```

"From any [of these] we can obtain 144 squares by first permuting the four columns in all 24 possible ways and then permuting the three rows other than the initial row in all 6 possible ways. Thus, there are altogether 4 . 144 = 576 distinct Latin squares of order 4."

"There are 56 different Latin squares of order 5 in which the initial row and column are in the natural order, each giving rise to 5! . 4! = 2,880 different squares by permuting the columns and then the rows other than the initial row. Thus there are 161,280 distinct Latin squares of order 5."

1987 -- Stein

Stein, M. 1987. Large Sample Properties of Simulations Using Latin Hypercube Sampling. *Technometrics.* 29(2): 143-151.

"Latin hypercube sampling (McKay, Conover, and Beckman 1979) is a method of sampling that can be used to produce input values for estimation of expectations of functions of output variables. The asymptotic variances of such an estimate is obtained. The estimate is also shown to be asymptotically normal."

1987 -- Kull and Specker

Kull, H. and E. Specker. 1987. Direct Construction of Mutually Orthogonal Latin Squares. *Computation Theory and Logic.* 224-236.

"The purpose of [algorithm] (DC) is to construct a system of mutually orthogonal latin squares ("MOLS" [3])."

1987 -- Massey, Maurer and Wang

Massey, J., U. Maurer and M. Wang. Non-Expanding, Key-Minimal, Robustly-Perfect, Linear and Bilinear Ciphers. *Advances in Cryptology -- EUROCRYPT '87.* 237-247. Springer-Verlag.

"Section 2 introduces the notion of a robustly-perfect block cipher and shows the connection of such ciphers to Latin squares."

"In a deterministic secret-key cipher, the ciphertext *Y* can be written in terms of the plaintext *X* and the key *Z* in the manner

Y = *f*(*X*,*Z*)

"Shannon [1,p.679] has defined a cipher system (*f*,P_Z) to be perfect if *X* and *Y* are statistically independent."

"Shannon observed that condition (2) of the above proposition *[proposition 1]* shows that the essential feature of a non-expanding key-minimal robustly-perfect cipher is that, with the rows indexed by plaintexts and the columns indexed by the keys, the array or corresponding ciphertexts forms a Latin square."

1987 -- Hunter

Hunter, J. 1987. Are Some Latin Squares Better Than Others? *Design, Data and Analysis.* 163-170.

Shows that order 3 Latin squares divide into two groups with different graph structures.

1989 -- Wolf

Wolf, M. 1989. Nondeterministic Circuits, Space Complexity and Quasigroups. *Computer Sciences Technical Report #870.* Computer Sciences Department, University of Wisconsin -- Madison.

"**Definition:** A *Latin square* is an *n* x *n* grid with each of the integers 1,2,...,*n* appearing exactly once in each row and column."

"If each of the integers 1,2,...,*n* appears as a label for exactly one row and exactly one column then the Latin square can be viewed as a multiplication table of a quasigroup. We formalize the definitions of groups and quasigroups by considering the following four properties of a set *Q* with an associated binary operation *. For all *a*,*b*,*c* in *Q*:"

1. There is a unique *x* such that *a***b*=*x*.
2. There is a unique *x* such that *a***x*=*b*.
3. There is a unique *x* such that *x***a*=*b*.
4. (*a***b*)**c*=*a**(*b***c*)

"**Definition:** *Q* is a *group* if * satisfies properties 1,2,3, and 4."

"**Definition:** *Q* is a *quasigroup* if * satisfies properties 1,2 and 3."

"Thus a quasigroup is more general than a group. In this paper we view quasigroups of order *n* as a binary function on {1,2,...,*n*}, thus the corresponding multiplication table is a Latin square. Viewing Latin squares *L* and *L'* as trinary relations <,,> and <,,>', *L* is *isomorphic* to *L'* if there exists a permutation *g* such that if <*x*,*y*,*z*> is in *L* then <*g*(*x*),*g*(*y*),*g*(*z*)>' is in *L'*."

1990 -- Godsil and McKay

Godsil, C. and B. McKay. 1990. Asymptotic Enumeration of Latin Rectangles. *Journal of Combinatorial Theory, Series B* 48: 19-44.

"A *k* x *n Latin rectangle* is a *k* x *n* matrix with entries from {1,...,*n*} with the property that no entry occurs more than once in any row or column. Thus an *n* x *n* Latin rectangle is nothing more than a Latin square."

"We prove that the number of *k* x *n* Latin rectangles" (this is *L*(*k*,*n*)) "is asymptotically

```
      k                          k n          -n/2   -k/2
 (n!)  ( n(n-1)...(n-k+1) / n  )  (1 - k/n)       e

                             (6/7)
as n approaches infinity, with k = o(n     )."
```

"The first attack on this problem was made by P. Erdos and I. Kaplanski [8], who showed that, for *k* = *O*(log *n*)$^{3/2 - e}$),

```
              k          k
  L(k,n) ~ (n!)   exp( -(   ) ).
                         2
```

"Further progress was made by Yamamoto [36] and Stein [27], who proved that

```
              k          k      3
  L(k,n) ~ (n!)   exp( -(   ) - k  / 6n )
                         2
```

for *k* = *O*($n^{5/12 - e}$) and *k* = *o*($n^{1/2}$).

1991 -- Byers

Byers, J. 1991. Basic Algorithms for Random Sampling and Treatment Randomization. *Computers in Biology and Medicine.* 21(112): 69-77.

"Five BASIC programs to select random samples from populations or to randomize treatments are presented." "Program 2 produces Latin squares of any size for treatment randomization."

"The algorithm uses the method of randomizing numbers by treatment . . . for two such arrays of treatments (columns and row). The column and row arrays . . . can then be used to calculate a unique latin square of . . . cell elements. Each column and row intersection, cell(c,r), of the latin square can be obtained from the sum of the numbers in the respective column and row arrays . . . ."

1991 -- Denes and Keedwell

Denes, J. and A. Keedwell. 1991. *Latin Squares: New Developments in the Theory and Applications.* North-Holland.

"A latin square of order n is an n x n matrix L whose entries are taken from a set S of n symbols and which has the property that each symbol from S occurs exactly once in each row and exactly once in each column of L."

"A set S on which a binary operation (.) is defined forms a quasigroup with respect to that operation if, when any two elements a,b of S are given, the element a.b is in S and the equations a.x = b and y.a = b each have exactly one solution in S."

"It is easy to see that the multiplication table (often called Cayley table)of a quasigroup is a latin square."

"Two latin squares $L_1 = ||a_{ij}||$ and $L_2 = ||b_{ij}||$ on n symbols, say 1,2,...,n, are said to be orthogonal if every ordered pair of symbols occurs exactly once among the n^2 pairs (a_{ij},b_{ij}), i = 1,2,...,n; j = 1,2,...,n."

"A pair of orthogonal latin squares is also called a graeco-latin square, especially in statistical applications, because L. Euler (1779) used Greek letters for one square of the pair and latin letters for the other."

1991 -- Camion, Carlet, Charpin and Sendrier

Camion, P., C. Carlet, P. Charpin and N. Sendrier. 1991. On Correlation-Immune Functions. *Advances in Cryptology -- CRYPTO '91.* 86-100. Springer-Verlag.

"In a general type of running-key generator, the output sequences of m Linear Feedback Shift Registers are taken as arguments of a singe non linear combining function f. If the function is not properly chosen, it can happen that the generator structure is not resistant to a *correlation attack* . . . ."

"The kth-order correlation immune functions were introduced by T. Siegenthaler in [6]."

"X. Guo-Zhen and J. L. Massey later gave an equivalent definition, using the Walsh transform of the boolean functions."

"In section 3, we first prove (theorem 1) that a k-ci function is an *orthogonal array of strength k* . . . ."

1992 -- Shao and Wei

Shao, J. and W. Wei. 1992. A formula for the number of Latin squares. *Discrete Mathematics.* 110: 293-296.

"A Latin square of order *n* is an *n* x *n* matrix, each row and column of which is a permutation of the set of letters I_n = {1,2,...,*n*}. A *k* x *n* Latin rectangle (*k* <= *n*) is a *k* x *n* matrix on the set I_n, whose rows are permutations of the letters {1,2,...*n*} and whose columns do not contain any repeated letter."

"We establish an explicit formula for the number of Latin squares of order *n*:

```
                          sigma0(A)    per A
  L[n] = n!   SUM(      (-1)          (       )
            A in B[n]                     n
```

where B[n] is the set of *n* x *n* (0,1) matrices, sigma0(A) is the number of zero elements in the matrix A and per A is the permanent of the matrix A."

"**Definition 1.** Let *m* <= *n* be integers, and S_n(m) the set of all *m*-permutations of the elements in the set In = {1,2,...,*n*}.

"For any *m* x *n* real matrix A = (a_{ij}), define the permanent per A of the matrix A to be

```
  per A =       SUM(   a[1,i1] a[2,i2] ... a[m,im]  ) .
           (i1,...,im) in Sn(m)
```

"For the special case *m* = *n* we have

```
  per A =       SUM(   a[1,i1] a[2,i2] ... a[n,in]  ) .
           (i1,...,in) in Sn(n)
```

"**Definition 2.** An *n* x *n* permutation matrix is a (0,1) matrix of order *n*, each row and column of which contains exactly one nonzero element." (Apparently this is the *n* x *n* (0,1) matrix described initially.)

1995 -- McKay and Rogoyski

McKay, B. and E. Rogoyski. 1995. Latin Squares of Order 10. *Electronic Journal of Combinatorics.* 2(3): 1-4.

"We describe two independent computations of the number of Latin squares of order 10. We also give counts of Latin rectangles with up to 10 columns, and estimates of the number of Latin squares of orders up to 15."

Numbers of normalized Latin squares (excerpted from Table 1: Numbers of normalized Latin rectangles)

n	L(n,n)
1	1
2	1
3	1
4	4
5	56
6	9,408
7	16,942,080
8	535,281,401,856
9	377,597,570,964,258,816
10	7,580,721,483,160,132,811,489,280

"To obtain the total number of Latin rectangles, not necessarily normalized, multiply L(k,n) by n!(n-1)!/(n-k)!."

Table 2. Estimates of L(n,n) for larger n.

n	trials	L(n,n)
11	1000000	5.36×10^{33}
12	1100000	1.62×10^{44}
13	400000	2.51×10^{56}
14	200000	2.33×10^{70}
15	20000	1.5×10^{86}

(*Editorial Comment: It is tempting to extrapolate this to order 16, by taking the ratios of adjacent values for L(n,n) as powers of 10, to get the sequence (11,12,14,16). If we then take the next element of the sequence as 18, we would expect L(16,16) to be something like* 10^{102}*, or about* 2^{339}.)

1996 -- Jacobson and Matthews

Jacobson, M. and P. Matthews. 1996. Generating uniformly distributed latin squares. *Journal of Combinatorial Designs.* 4(6): 405-437.

(*If you have access to this article, please send a copy to my current address so I can include excerpts here!*)

Terry Ritter, his current address, and his top page.

Last updated: 1998-05-29