Cryptography: RSA Encryption and Decryption

Greg Plaxton
Theory in Programming Practice, Spring 2004
Department of Computer Science
University of Texas at Austin
Joining the RSA Cryptosystem: Quick Review

• First, Bob randomly chooses two large (e.g., 512-bit) primes \( p \) and \( q \)

• Then, Bob computes \( n = pq \), \( \phi(n) = (p - 1)(q - 1) \), and a positive integer \( d < n \) such that \( d \) and \( \phi(n) \) are relatively prime
  – In particular, any prime exceeding \( \max(p, q) \) (and less than \( n \)) is a valid choice for \( d \)

• Then, Bob computes \( e \) such that \( de \) is congruent to 1 modulo \( \phi(n) \)

• Bob’s public key is \((e, n)\) and Bob’s private key is \((d, n)\)
RSA Encryption and Decryption

• Choose the highest block size $b$ such that every $b$-bit number is less than $n$
  – Thus $b$ is $\lfloor \log_2 n \rfloor$
  – For example, if $p$ and $q$ are 512-bit numbers, then $b$ is either 1022 or 1023

• Suppose Alice wants to send a message to Bob
  – She partitions the message into a sequence of $b$-bit blocks (padding the last block with zeros if necessary)
  – Encryption and decryption is done on a per block basis
  – Later we’ll discuss some variations of this basic framework
Encryption of a Single Block

• Suppose Alice wants to send message block $X$ to Bob
  – The message block $X$ is a $b$-bit string
  – We interpret $X$ as a nonnegative integer in the usual manner, e.g., if $X$ is the 5-bit string 00110 then we interpret $X$ as 6
  – By our choice of $b$, $X$ is less than $n$

• Alice encrypts $X$ by computing the number $Y$ equal to $X^e \mod n$; note that $Y$ is less than $n$ and thus has at most $b' = 1 + \lceil \log_2 (n-1) \rceil \leq b+1$ bits in its binary representation

• Alice sends $Y$ to Bob
  – Alice could send $Y$ as a $b'$-bit string (i.e., padded with leading zeros if necessary)
Decryption of a Single Block

- Bob receives encrypted message block $Y$ and would like to recover the corresponding plaintext message block $X$.

- Bob computes the number $Z$ equal to $Y^d \mod n$; note that $Z$ is less than $n$.

- We claim that $Z = X$.
  - Lemma: For any integers $a$ and $b$, and any positive integer $c$, $(ab) \mod c$ equals $((a \mod c)b) \mod c$.
  - It follows that $Y^d \mod n$ is equal to $X^{de} \mod n$.
  - It remains to prove that $X^{de} \mod n$ equals $X$.

Theory in Programming Practice, Plaxton, Spring 2004
Lemma: \(X^{de} \mod p\) equals \(X \mod p\)

- Recall that \(e\) was chosen so that \(de\) is congruent to 1 modulo \(\phi(n) = (p - 1)(q - 1)\)

- Thus \(de = t(p - 1) + 1\) for some nonnegative integer \(t\)

- Thus \(X^{de} \mod p\) equals

\[
\left[(X^{p-1} \mod p)^t \cdot X\right] \mod p
\]

- By Fermat’s Little Theorem, \(X^{p-1} \mod p\) is equal to 1 for \(X \neq 0\) (if \(X = 0\), the lemma holds trivially)

- Hence \(X^{de} \mod p\) equals \(X \mod p\), as desired
**Theorem:** $X^{de} \mod n$ equals $X$

- We have just established that $X^{de} - X$ is a multiple of $p$.
- A symmetric argument shows that $X^{de} - X$ is a multiple of $q$.
- Thus $X^{de} - X$ is a multiple of $n$, i.e., $X^{de}$ is congruent to $X$ modulo $n$.
- The claim of the theorem follows since $0 \leq X < n$. 

Theory in Programming Practice, Plaxton, Spring 2004
Modular Exponentiation

• It remains to show how to compute $a^b \mod c$ efficiently

• The naive approach is to compute $a^2, a^3, a^4, \ldots, a^b$ and then compute the remainder when the last number in this sequence is divided by $c$
  
  – If $b$ is a 512-bit number, say, the length of this sequence is astronomical
  
  – Furthermore, the length of each number in the last half, say, of this sequence is astronomical

• A slightly less naive approach is to observe that we can compute $a \mod c, a^2 \mod c, a^3 \mod c, a^4 \mod c, \ldots, a^b \mod c$
  
  – This ensures that we are always working with numbers in the range $\{0, \ldots, c - 1\}$
  
  – However, the length of the sequence remains astronomical
Fast Exponentiation

• Suppose we want to compute $a^b$, where $a$ and $b$ are nonnegative integers, using a small number of multiplications
  – For the moment, let us ignore any difficulties associated with multiplying astronomically large numbers
  – We’ll simply charge one unit of time for each multiplication

• What is an efficient way to compute $a^b$ when $b$ is of the form $2^k$ for some nonnegative integer $k$?

• What about the case of general $b$?
**Fast Exponentiation by Repeated Squaring**

- **Example:** Suppose we want to compute $a^b$ where $b = 35 = 100011_2$
- We can compute $a^2$, then $a^4$, then $a^8$, then $a^{16}$, then $a^{17}$, then $a^{34}$, then $a^{35}$
  - Note that $2 = 10_2$, $4 = 100_2$, $8 = 1000_2$, $16 = 10000_2$, $17 = 10001_2$, $34 = 100010_2$, $35 = 100011_2$
- It is often more convenient to examine the bits of $b$ starting with the low order position and to compute, e.g., $(a, a)$, $(a^2, a^3)$, $(a^4, a^3)$, $(a^8, a^3)$, $(a^{16}, a^3)$, $(a^{32}, a^{35})$
  - As above, we use a total of seven multiplications
  - At each iteration, we examine the low-order bit of $b$ and then shift $b$ right (dropping the low order bit)
  - The loop terminates when $b$ is zero

Theory in Programming Practice, Plaxton, Spring 2004
Fast Modular Exponentiation

• To compute $a^b \mod c$, we proceed as on the previous slide (either method will work), but every time we compute a product we take the result modulo $c$

• Example: Suppose we want to compute $11^{35} \mod 13$

• Using the first method from the previous slide, we compute $11^2 \mod 13 = 4$, $11^4 \mod 13 = 4^2 \mod 13 = 3$, $11^8 \mod 13 = 3^2 \mod 13 = 9$, $11^{16} \mod 13 = 9^2 \mod 13 = 3$, $11^{17} \mod 13 = 3 \cdot 11 \mod 13 = 7$, $11^{34} \mod 13 = 7^2 \mod 13 = 10$, $11^{35} \mod 13 = 10 \cdot 11 \mod 13 = 6$

• Using the second method, we compute $(11, 11)$, $(4, 5)$, $(3, 5)$, $(9, 5)$, $(3, 5)$, $(9, 6)$, so once again we get 6 as the answer
Performance of RSA

- A trick that is often used to speed encryption (but not decryption) is to choose \(d\) and \(e\) so that \(e\) is small.

- RSA encryption and decryption is quite fast, but not sufficiently fast for many high-speed network applications.
  - Accordingly, RSA is often only used to exchange a secret key.

- This secret key is not a one-time pad of the sort we discussed earlier in a previous lecture.
  - Recall that such a one-time pad would have to be as large as the message we intend to transmit.

- Instead, the secret key is often used to determine a block cipher encryption of the data.
A block cipher is a function that takes two inputs, a plaintext block and a key, and produces as output a ciphertext block.

- The plaintext and ciphertext blocks are normally of the same size (e.g., 64 bits is common).
- The key may be a different size; in practice, it is often 64 or 128 bits.

A good block cipher must satisfy the following properties:

- Given the key and the plaintext (resp., ciphertext) block, it is easy for a computer program to determine the corresponding ciphertext (resp., plaintext) block.
- Given a plaintext block $M$ and the corresponding ciphertext block $C$, it is computationally hard to determine a key mapping $M$ to $C$. 
Block Cipher Encryption Modes

• Assume that the sender and receiver have agreed on a block cipher and a secret key

• Electronic codebook encryption mode: Just divide the message into blocks and apply the block cipher to each block
  – A serious disadvantage of this scheme is that multiple copies of the same plaintext block all map to the same ciphertext block

• Cipher block chaining encryption mode:
  – The first ciphertext block is computed as above
  – For $i > 1$, the $i$th ciphertext block is obtained by applying the block cipher to the XOR of the $i$th plaintext block and the $(i - 1)$th ciphertext block
  – How do we decrypt in this case?

• Other encryption modes exist