Introduction à la cryptologie et aux cryptosystèmes

- **Cryptologie** – La *cryptologie* regroupe la *cryptographie*, qui est la science de garder secret de messages et la *cryptanalyse*, qui est l'art de décrypter des messages chiffrés.

Cryptographie

- **Protocoles** – Exemples :
  - Protocole d'échange de message ;
  - Preuve avec divulgation nulle ;
  - Identification ;
  - Mise en gage ;
  - Vote électroniques…

- **Protocole d'échange de message** – On recherche plusieurs critères :
  - *la confidentialité* ;
  - *l'authentification*, c'est-à-dire que le destinataire d'un message doit pouvoir s'assurer de son origine ;
  - *l'intégrité*, c'est-à-dire que le destinataire doit pouvoir s'assurer de la non modification du message ;
  - *la non répudiation ou le non désaveu*, c'est-à-dire que l'expéditeur ne doit pas pouvoir nier avoir envoyer ce message.

- **Système cryptographique** – Un système cryptographique est un n-uplet (M, C, K, E, D) où :
  - M est l'ensemble des textes clairs possibles ;
  - C est l'ensemble des textes chiffrées possibles ;
  - K est l'ensemble des clefs possibles ;
  - pour tout $k \in K$, il y a une règle de chiffrement dans E, $e_k : M \to C$ et une règle de déchiffrement dans D, $d_k : C \to M$ telles que $\forall m \in M, d_k(e_k(m)) = m$.

- **Système à clef secrète ou privée** – On peut calculer facilement d_k en fonction de e_k, le plus souvent même $d_k = e_k$. Ce cryptosystème est rapide, les clefs sont courtes. Mais, il nécessite une clef par couple de personnes, un tiers de confiance et un canal sûr pour l'échange des clefs. Exemple : DES.

- **Système à clef publique** – e_k est publique, d_k est privée. Impossibilité de calculer « facilement » d_k d'après e_k. Ce cryptosystème nécessite une clef par personne. Mais il est plus lent en comparaison avec le système à clef privée, et les clefs sont plus longues. Exemple : RSA.

- **Protocoles hybrides** : Une idée consiste à utiliser un système à clef publique pour transmettre une clef de session, et un système à clef secrète pour crypter le texte clair. Exemple : PGP.

Cryptanalyse

- **Attaquant** – Un attaquant a plein accès aux communications entre l'expéditeur et le destinataire. Par ailleurs, il connaît tous les détails de l'algorithme.

- **Attaque d'un cryptosystème** – Tentative de cryptanalyse. On distingue différents types d'attaque d'un cryptosystème en fonction des données supposées connues par l'attaquant.
  - *Attaque à texte chiffrée,* où l'attaquant connaît plusieurs messages cryptés $e_k(m_i)$ (situations courantes) ;
  - *Attaque à texte en clair connu,* où l'attaquant connaît plusieurs couples message en clair – message crypté : $(m_i, e_k(m_i))$;
  - *Attaque à texte en clair choisi statique ou adaptative,* où l'attaquant peut connaître des couples $(m_i, e_k(m_i))$ de son choix, soit au début (attaque statique), soit quand il le souhaite (attaque adaptative) ;
  - *Attaque à texte chiffré choisi,* où l'attaquant peut connaître des couples $(c_i, d_k(c_i))$ de son choix.

- **Attaque exhaustive** – Une attaque *exhaustive* essaie toutes les fonctions de décryptage possible en parcourant l'espace des clefs.

- **Attaque réussie d'un cryptosystème –** On distingue plusieurs niveaux de réussite selon la connaissance que l'on obtient du cryptosystème :
  - *le cassage complet,* où l'attaquant trouve la clé *k* et donc d_k ;
  - *l'obtention globale,* où l'attaquant trouve d_k par un algorithme donnant $d_k(c)$ pour tout *c* ;
  - *l'obtention locale,* où l'attaquant trouve $d_k(c)$ pour un message *c* intercepté ;
  - *l'obtention d'information* sur le texte en clair ou la clef.

- **Sécurité d'un cryptosystème**
  - *Sécurité inconditionnelle* **:** Un cryptosystème est *inconditionnellement sûr* si pour tous *x* et *y*, $P(m = x | c = y) = P(m = x)$. Un tel cryptosystème ne peut pas être cassée même avec une puissance de calcul infinie. *Shanon* a prouvé qu'une telle sécurité nécessitait que la longueur de la clef égale au moins celle du texte chiffré. Exemple *du chiffrement de Vernam.*
  - *Sécurité calculatoire* : Un cryptosystème est sûr au sens de la théorie de la complexité si le meilleur algorithme pour le casser est « trop complexe » en temps ou en espace. Malheureusement nous ne savons pas si P≠PN !
  - *Sécurité calculatoire conditionnelle* : Si tout algorithme connu ou non « est supposé » trop complexe… si le meilleur algorithme connu…
  - *Sécurité Ad Hoc* : Si le coût pour casser le système est supérieur au gain espéré.

Chiffrement par blocs

- **Principe** – Étant donné un cryptosystème (M, C, K, E, D), avec $M = C = A^n$ et $A = \{0,1\}$ le plus souvent ; on décompose toute séquence $m \in A^*$ en *p* blocs de A^n, ce qui donne $m = m_1 \cdots m_p$, puis on crypte selon $c = e_k(m_1) \cdots e_k(m_p)$.

- **Exemples** – ECB, OFB, CFB, CBC…

- **Produit de cryptosystèmes –** (…) Dans le but de renforcer un cryptosystème, en élargissant l'espace des clefs.

- **Chiffrement par décalage**[1] **–** (M,C,K,E,D) où $M=C=(Z_{26})^n$, $K=[0,25]$, $c=e_k(m)=(m_1+k,\cdots,m_p+k) \bmod 26$ et $m=d_k(c)=(m_1-k,\cdots,m_p-k) \bmod 26$. Une attaque exhaustive suffit, le nombre de clefs étant faible !

- **Chiffrement affine –** (…)

- **Chiffrement par substitution –** (M,C,K,E,D) où $M=C=[a-z]^n$, K est l'ensemble des permutations sur $[a-z]$, $c=e_k(m)=k(m)$ et $m=d_k(c)=k^{-1}(c)$.

- **Attaque du chiffrement par substitution** – Une attaque exhaustive à texte chiffré n'est pas envisageable, le nombre de clefs étant *26!*. En revanche, un attaque exhaustive à texte clair est possible dans une certaine mesure. Il existe d'autres méthodes plus subtiles, basées sur la fréquence de quelques séquences littérales dans la langue anglaise, permettant de mener une attaque à texte chiffrée.

- **Chiffrement de *Vigenère* (16ème siècle) –** (M,C,K,E,D) où $M=C=[a-z]^p$, $K=[a-z]^p$, c'est-à-dire que l'on travaille sur des blocs m_i de taille p et des clefs de taille aussi p, d'où $p=\lfloor\sqrt{|m|}\rfloor+1$; $c=e_k(m)=(m_1+k_1,\cdots,m_p+k_p) \bmod 26$ et $m=d_k(c)=(m_1-k_1,\cdots,m_p-k_p) \bmod 26$.

- **Attaque du chiffrement de *Vigenère*** – On estime p avec *le test de Kasiski* (deux sous-séquences de m ont même codage si leur décalage est égale à *0 mod p*) ou avec *le test des indices de coïncidence $I_c(x,y,d)$*. L'indice de coïncidence représente la probabilité qu'un caractère du texte x plus un décalage d soit égal au caractère du texte y. Grâce à des considérations statistiques sur la langue anglaise, on en déduit la taille du bloc p. On estime les clefs $k_1,\cdots,k_p$ toujours avec *les indices de coïncidence mutuels*. On déduit m.

- **Réseau de *Feistel* à *n* rondes**[2] **–** (M,C,K,E,D) où $M=C=(Z_2)^{21}$. Soit une fonction quelconque $f:Z_2\times K\to Z_2$. On génère une séquence de n sous-clefs $k_1,\cdots,k_n$ à partir de la clef k.
  - $m=L_0R_0$
  - une ronde de *Feistel* calcule L_iR_i à partir de $L_{i-1}R_{i-1}$ selon $L_i=R_{i-1}$ et $R_i=L_{i-1}+f(R_{i-1},k_i)$;
  - $c=L_nR_n$.

  Le décryptage se fait par le même réseau de *Feistel* en inversant l'ordre des sous-clefs. La plupart des chiffrements symétriques sont construits sur des réseaux de *Feistel* ou leurs extensions (DES, IDEA…).

- **DES –** Le DES est un réseau de *Festeil* à 16 rondes, à clef k de 56 bits diversifié en 16 clefs de 48 bits, codant des blocs de 64 bits. Le texte en clair est préalablement

[1] $Z_m=[0,m-1]$ en bijection avec l'alphabet.

[2] Le + désigne l'addition modulo 2, c'est-à-dire le ou exclusif.

transformé selon la permutation *IP*, puis le texte issu des 16 rondes de *Feistel* est finalement transformé selon la permutation IP^{-1}.

- diversification de la clef selon un protocole établi…
- fonction *f* défini…

Fonctions à sens unique, Problèmes difficiles en théorie des nombres

- **Fonction à sens unique –** Une fonction à sens unique $f : M \to C$ est telle que $\forall x \in M$, $f(x)$ est facile à calculer et $\forall y \in C, y = f(x)$, calculer *x* est difficile.

- **Fonction à sens unique à brèche secrète –** La fonction est à brèche secrète si il existe une information *k*, telle qu'étant donné *k* et *y*, il devient facile de calculer tous les *x* tels que $y = f(x)$.

- **Cryptosystème à clef publique –** Soit $f : M \to C$ une fonction à sens unique et à brèche secrète *k*. La clef publique est *f*. La clef secrète est (f, k).

  Alice calcule f(m).
  Alice envoie c = f(m) à Bob.
  Bob calcule $f^{-1}(c)$ *en utilisant la brèche secrète k.*

- **Fonction de hachage à sens unique –** Toutes les images de *m* par *f* ont mêmes longueurs.

- **Fonction à collision faible et forte –** Dans les deux cas, $\forall x \in M$, $f(x)$ est facile à calculer. De plus, une fonction à faible collision est telle qu'il est difficile de calculer un $x' \in M$ tel que $f(x') = f(x)$. Pour une fonction à forte collision, il est difficile de calculer *x* et *x'* tels que $f(x') = f(x)$.

- **Classes P, NP, NP complet, FP, FNP…** – La taille d'un mot *m* est noté $|m|$.
  - *Classe P* : Classe des problèmes « faciles », résoluble en un temps polynomiale fonction de la taille des mots.
  - *Classe NP* : Classe des problèmes à vérification « facile », dont la vérification s'effectue en temps polynomiale de la taille des mots. Par exemple, vérification qu'un graphe est 3-coloriable.
  - *Classe NP complet* : Classe des problèmes aussi « durs » que chacun des *NP*. Par exemple, 3-coloriage.
  - *Classe FP et FNP* : La classe des problèmes fonctionnels associés à *NP* est *FNP*. Ceux résolubles en temps polynomial est *FP*. Une fonction à sens unique est dans *FP*, mais sa réciproque est dans *FNP*.
  - *Classe RP* : Classe des problèmes résolubles par un algorithme probabiliste (dont toute réponse positive est correcte et dont toute réponse négative est correcte avec une probabilité *e*.
  - *Classe ZPP* : Classe des problèmes résolubles par un algorithme de *Las Vegas* (algorithme de reconnaissance d'un langage L en temps polynomiale par rapport à la taille du mot en entrée).

- **Problèmes difficiles –** *Factorisation, problème du sac à dos, inverse, racines, logarithme discret dans Z_n[3] ou Z_n^*[4], résiduosité quadratique, racine carré modulaire, RSA, Diffie-Hellman…* Ces problèmes sont considérés insolubles à partir de 512 chiffres !
  - La factorisation d'un entier *n* est le couple (p,q) d'entiers premiers tels que $pq = n$.
  - L'inverse d'une entier *a* modulo *n* est un entier *b* tel que $ab = 1 \bmod n$.
  - Un entier *x* de *Jn* (ensemble de Jacobi) est un résidu quadratique s'il est de la forme $a^2 \bmod n$.
  - Le logarithme discret modulo *n* d'un entier *y* selon un générateur[5] *g* de Z_n^* et un nombre premier *p*, est un entier *e* (le plus petit) tel que $g^e = y \bmod n$.
  - RSA des entiers *n*, *e* et *c* avec *n=pq*, *e* premier avec *(p-1)(q-1)*, *p* et *q* premiers, est *m* tel que $m^e = c \bmod n$.
  - *Diffie-Hellman* de *p*, un entier premier, de *g*, un générateur de Z_n^* et de deux entiers $g^a \bmod p$ et $g^b \bmod p$, est l'entier $g^{ab} \bmod p$.

Quelques algorithmes sur les entiers, complexités et théorèmes

- **Taille d'un entier –** La taille d'un entier *n*, noté $|n|$, est $\lceil \log_2(n) \rceil$.

- **Complexité en temps –** La complexité en temps des algorithmes est étudié par rapport à la taille des entrées.

- **Règles de calcul avec modulo**
  - $p \times q \bmod k = p \bmod k \times q \bmod k$;
  - binôme de *Newton* : $(x \bmod y)^z \bmod y = x^z \bmod y$;

- **Théorème *d'Euler*** – Si *x* est premier avec *p* alors $x^{\phi(p)} = 1 \bmod p$. Si *p* est premier et $x \in [0, p-1]$, alors $\phi(p) = p-1$ et $x^{p-1} = 1 \bmod p$. On rappelle que $\phi(p)$ est le nombre de nombre premiers avec *p* et inférieur ou égale à *p*.

(…)

Primalité

- **Primalité –** Teste si un entier *n* est premier.

- **Co-Primalité** – Teste si un entier *n* n'est pas premier.

(…)

[3] Z_n est l'ensemble des entiers naturels inférieur strictement à *n*.
[4] Z_n^* est l'ensemble des entiers naturels premiers inférieur à *n*.
[5] Un générateur *g* d'un groupe fini *G* est un élément de *G* tel que ses puissances successives engendrent *G*.

Preuves à divulgation nulle

- **Définition –** Un protocole de preuve à divulgation nulle est un protocole entre deux machines (dites le *prouveur* et le *vérificateur*) portant sur la résolution d'un problème P et qui vérifie *la consistance, la significativité, la divulgation nulle*.

- **Honnêteté –** Un parti est honnête si il respecte scrupuleusement son protocole.

- **Consistance** – On suppose le prouveur et le vérificateur honnêtes. Si l'entrée est une instance positive du problème P, alors le protocole permet au vérificateur d'en être persuadé.

- **Significativité –** On suppose le prouveur malhonnête et le vérificateur honnête. Si l'entrée est une instance négative du problème, alors la probabilité que le vérificateur soit trompée peut être rendu aussi petite que voulu : $P(\textit{output is OUI}) < 2^{-t}$ où t est fixé par le vérificateur lui-même. Ainsi si le prouveur ne sait pas résoudre le problème et donc essaie de tromper le vérificateur, ce dernier s'en aperçoit avec une forte probabilité.

- **A divulgation nulle** – On suppose le vérificateur malhonnête (il ne choisit pas ses données aléatoirement, par exemple, pour tenter d'obtenir des informations sur la preuve) et le prouveur honnête. A la fin du protocole, le vérificateur ne sait pas résoudre le problème P (ou du moins, pas davantage qu'avant le protocole).

- **Non isomorphisme de graphes** – (…)

- **Isomorphisme de graphes** – (…)

- **3-coloriabilité** – (…)

- **Mise en gage** – Un gageur souhaite prouver à une adversaire qu'il peut donner une prédiction, mais ne souhaite pas la révéler immédiatement. Protocole en 2 étapes.
  - La *mise en gage*, à proprement parler : le gageur écrit ses informations dans un coffre fort à deux clefs différentes, conserve une clef et en donne une à son adversaire.
  - Puis, la *révélation* : quand les deux parties sont d'accord, ils ouvrent le coffre avec leurs deux clefs, l'adversaire lit le message.

  Ainsi l'adversaire ne peut lire le message sans l'autorisation du gageur, et le gageur ne peut le modifier sans que son adversaire s'en aperçoive.

Identification, Signature

- **Protocole d'identification** – L'identification de *x* à *y* se déroule en deux temps.
  - Fournir à *y* un problème P et le convaincre que la seule personne qui puisse le résoudre est *x*.
  - Résoudre le problème lui-même.

  Le protocole d'identification étant supposé écouté, *x* souhaite empêcher quiconque de se faire passer pour lui. *y* souhaite que personne ne s'identifie en tant que *x*.

- **Certification par une autorité de confiance** – Une autorité de confiance (*trust authority*) a par définition la définition de tous les participants.
  - *x* envoie à *TA (ID(x), P)* avec *P* un problème

- *TA* vérifie que nul n'est identifié par *P*
- *TA* envoie le message signe appelé certificat de *x, c(x).*
- L'identification se fait simplement par envoie de *c(x)* à *y ; x* convainc *y* qu'il sait résoudre *P*.

- **Procédé de signature** – (…)

Vote

(…)

Distribution de clefs

(…)