MODULE ID12 : INTRODUCTION À LA CRYPTOLOGIE
COURS MAGISTRAL 1

PAUL ZIMMERMANN

L'objet du module est d'introduire les problématiques et les outils de base de la cryptologie, et d'entreprendre une étude détaillée de ces derniers dans le cas de la cryptologie symétrique (à clé secrète) et asymétrique (à clé publique).

1. Problématique de la cryptologie

La *cryptographie* consiste à trouver des procédés de protection de l'information (chiffrement, signature, ...)

La *cryptanalyse* consiste au contraire à essayer de « casser » les procédés mis au point par le cryptographe.

La *cryptologie* regroupe ces deux disciplines, cryptographie et cryptanalyse.

Ce module est largement inspiré du livre *Handbook of Applied Cryptography* de Menezes, van Oorschot et Vanstone (CRC Press, 1997), `http://www.cacr.math.uwaterloo.ca/hac/`.

2. Historique

Longtemps domaine réservé des militaires, des diplomates ou des gouvernements en général, la cryptologie a pris un nouvel essor ces dernières années avec l'avénement du « commerce électronique », grand demandeur de primitives sûres. L'adoption en 1977 du standard DES (*Data Encryption Standard*) par le FIPS (*Federal Information Processing Standard*) est symbolique de cette évolution.

L'invention en 1976 — par Diffie et Hellman — de la cryptographie à clé publique crée une petite révolution pour l'époque. Il fallut cependant attendre deux ans, soit 1978, pour que Rivest, Shamir et Adleman mettent au point la première réalisation pratique de chiffrement à clé publique, avec le système maintenant bien connu sous le nom de RSA. La sécurité de RSA étant basée sur la difficulté de la factorisation d'entier, cela a relancé les recherches sur ce dernier problème. En 1985, El Gamal invente un autre procédé à clé publique, basé sur le problème du logarithme discret, qui avait déjà été utilisé par Diffie et Hellman. On voit donc déjà des liens étroits entre procédés cryptographiques et problèmes mathématiques.

Il faut cependant attendre 1991 pour que soit adopté le premier standard de signature à clé publique (ISO/IEC 9796), basé sur RSA, suivi en 1994 par un autre standard basé sur El Gamal.

3. Taxonomie des primitives

Le problème le plus connu en cryptologie est le *chiffrement*, à savoir comment transmettre un message d'Alice à Bob sans que le méchant Charlie ne puisse le lire. Le chiffrement n'est cependant qu'un des nombreux problèmes, ou *primitives*, que l'on peut rencontrer :

Date: 27 septembre 2005. Email : `zimmerma@loria.fr`.

- la *confidentialité* consiste à garder une information secrète, sauf de ceux qui sont autorisés à la connaître. Exemple : le responsable du module de cryptologie transmet les notes brutes de son module à la secrétaire, sans que les étudiants ne puissent y avoir accès, car ces notes seront peut-être modifiées pour qu'ils aient la moyenne ;
- l'*intégrité des données* consiste à protéger des données contre des altérations, volontaires ou non. Exemple : les notes recues par la secrétaire du Master doivent être les mêmes que celles envoyées par le responsable du module de cryptologie. Il ne faut pas non plus qu'une note de 19 se transforme par erreur en 09 du fait d'un bug de Windows ;
- l'*authentification* d'une personne ou d'une entité consiste à s'assurer de l'identité de cette personne ou entité. Exemple : le porteur d'un passeport doit prouver qu'il est bien le titulaire de ce passeport, par exemple par contrôle visuel (photo) ou empreinte digitale ;
- on distingue l'*authentification* d'un message, qui consiste à faire le lien entre ce message et son expéditeur. Exemple : authentification des communiqués envoyés par les groupes terroristes aux média ;
- la *signature* est un moyen de lier une information ou un document à une entité. Exemple : la signature papier ou la signature électronique ;
- le *contrôle d'accès* est un moyen de limiter l'accès de certaines ressources à certaines entités privilégiées. Exemple : les droits d'accès aux fichiers sous Unix ;
- la *certification* est la détention d'information par une autorité reconnue, un « tiers de confiance ». Exemple : Alice ne fait pas confiance en Charlie, mais Bob en qui elle a confiance lui dit que Charlie est irréprochable ;
- le *timestamping* permet de dater la création ou l'existence d'une information. Exemple : Alice a trouvé un algorithme polynomial de factorisation d'entier, veut pouvoir prouver qu'elle l'a trouvé en septembre 2005. Elle se fait donc prendre en photo au « Livre sur la place » ;
- le *marquage électronique*, par exemple modifier de façon invisible une image électronique, de manière à identifier des copies pirates ;
- la vérification de création ou d'existence d'une information par une personne autre que le créateur de cette information, sans forcément dévoiler cette information. Exemple : le responsable du module de cryptologie interroge Alice pour savoir si elle a bien compris l'algorithme RSA, sans lui faire dévoiler le détail de l'algorithme, afin d'interroger ensuite son voisin Bob ;
- l'*accusé de réception* permet de vérifier qu'une information est bien arrivée à son destinataire. Exemple : lettre ou mail avec accusé de réception ;
- l'*anonymat* permet de garder secrète l'identité d'une personne jouant un rôle dans un processus. Exemple : rapporteur d'un article, dénonciation d'un tricheur ;
- la *non répudiation* empêche une entité de nier des actions passées. Exemple : compromis de vente. La plupart du temps, un tiers de confiance est impliqué (exemple : notaire).

À chaque problème ou primitive correspond une ou plusieurs solutions, plus ou moins satisfaisantes. Une solution peut en général se présenter sous la forme d'un *protocole*, à savoir une suite d'actions — le plus souvent des messages — envoyés entre deux entités. Plus il y a d'entités ou de participants, plus le protocole est compliqué. Par exemple le *vote électronique* implique n participants, n pouvant être très grand, de l'ordre de plusieurs millions pour une élection nationale.

La sécurité d'un protocole dépend le plus souvent d'un modèle donné de l'attaquant, à savoir le méchant Charlie. Par exemple, certaines « preuves » de sécurité s'effondrent avec l'attaque dite *man-in-the-middle*, à savoir quand Charlie intercepte les messages envoyés entre Alice et Bob.

On peut cependant regrouper les primitives en trois grandes classes :
- les primitives sans clé (fonctions de hachage, permutations, suites aléatoires) ;
- les primitives symétriques, ou à clé privée (chiffrement symétrique, fonction de hachage, signature, suite pseudo-aléatoire, identification) ;
- les primitives asymétriques, ou à clé publique (chiffrement, signature, identification).

Le choix de telle ou telle primitive doit être fait en considérant un certain nombre de critères :
- le niveau de *sécurité* bien sûr. Il est souvent difficile à quantifier. Par exemple pour évaluer les primitives basées sur RSA, donc sur la factorisation d'entier, on considère le temps nécessaire aux meilleurs algorithmes actuellement connus (le record actuel est de 200 chiffres, établi en mai 2005). Pour trouver une clé de n bits, une borne triviale est de 2^n essais.
- la *fonctionnalité*. Une primitive de signature n'est pas forcément adaptée au chiffrement, et vice versa.
- le *mode opératoire*. Par exemple, le procédé RSA peut servir au chiffrement ou à l'authentification suivant la façon dont on s'en sert.
- la *performance*. Par exemple, on peut mesurer le nombre d'octets par seconde que permet de chiffrer telle ou telle primitive.
- la *facilité d'implantation*. Un algorithme effectuant des « ou » logiques sur des mots-machine (32 ou 64 bits) sera plus facile à implanter qu'un autre utilisant de l'arithmétique en précision arbitraire, ou encore des calculs sur le corps fini à 16 éléments.

La plupart du temps, on fait un compromis entre ces différents critères. Exemple : en cas de faible puissance de calcul (carte à puce par exemple), on accepte un niveau plus faible de sécurité, pour avoir un système plus rapide, donc mieux accepté par les utilisateurs. Exemple typique : un délai de plus de 3 secondes est redhibitoire pour l'identification via carte à puce.

4. Concepts mathématiques de base et terminologie

Fonction d'un ensemble X vers Y. Injection, surjection, bijection.

Fonction à « sens unique » : une fonction $f : X \to Y$ est dite à *sens unique* s'il est facile de calculer $f(x)$ pour $x \in X$, mais difficile pour $y \in Y$ de trouver $x \in X$ tel que $f(x) = y$.

Exemple : $f(x) = x + 1$ de $\mathbb{N}$ dans $\mathbb{N}$ n'est pas à sens unique. $f(x) = 3x \bmod 17$ de $\mathbb{Z}/(17\mathbb{Z})$ dans lui-même ne l'est pas non plus. Par contre, la fonction $f(x) = x^3 \bmod n$ avec $n = 3211286527$ est un exemple de fonction à sens unique (trouver par exemple une pré-image de $y = 3$).

Fonction à trappe : une fonction à sens unique est dite *à trappe* si avec une information supplémentaire — la trappe — il devient facile de trouver x tel que $f(x) = y$. Exemple : connaissant la factorisation $n = pq$ de n, il est facile de trouver les solutions de $x^3 = y \bmod n$, car il suffit de combiner par restes chinois celles de $x^3 = y \bmod p$ et de $x^3 = y \bmod q$, qui quant à elles s'obtiennent par factorisation du polynôme $x^3 - y$ sur $\mathbb{F}_p[x]$ ou $\mathbb{F}_q[x]$ (Cantor-Zassenhaus).

On désigne usuellement par $\mathcal{A}$ un **alphabet**, à savoir un ensemble de symboles. Par exemple $\mathcal{A} = \{0, 1\}$, ou les 26 lettres de l'alphabet.

$\mathcal{M}$ désigne l'espace des messages clairs, c'est un ensemble de mots dans un alphabet $\mathcal{A}$. Un élément m de $\mathcal{M}$ est appelé un *texte clair*.

$\mathcal{C}$ désigne l'espace des messages chiffrés, sur un alphabet $\mathcal{A}'$. Un élément c de $\mathcal{C}$ est un *texte chiffré*.

On note $\mathcal{K}$ l'espace des clés. Une clé $e \in \mathcal{K}$ définit de manière unique une bijection de $\mathcal{M}$ vers $\mathcal{C}$, notée E_e. On note D_d une bijection de $\mathcal{C}$ vers $\mathcal{M}$ de clé d. Un schéma de chiffrement est tel que $D_d(E_e(m)) = m$ pour tout m de $\mathcal{M}$. On a parfois $d = e$ et $D_d \equiv E_e$ (involution).

Règle d'or 1. La sécurité d'un procédé cryptographique ne doit pas reposer sur la confidentialité du *procédé* — ici la fonction de chiffrement employée, les ensembles $\mathcal{M}, \mathcal{C}, \mathcal{K}$ — mais sur la confidentialité de la *clé*.

Schéma classique de chiffrement à clé secrète : Alice et Bob commencent par choisir des clés secrètes (e, d). Quand Alice veut envoyer un message m à Bob, elle calcule $c = E_e(m)$, et envoie c à Bob. Bob reçoit c, et calcule $D_d(c) = m$.

L'intérêt d'un procédé à clé est que si la clé est compromise, il suffit de la changer, sans avoir à implanter un nouveau procédé. Analogue avec un cadenas à chiffres.

Comme la plupart des messages passent par un certain médium (Internet, téléphone, WiFi), on formalise la notion de *canal* :

- un *canal* est un moyen de transporter une information d'une entité à une autre ;
- un canal est dit *sûr* s'il n'est pas accessible au méchant Charlie, qui ne peut ni écouter, ni insérer, ni supprimer, ni réordonner ce qui passe sur ce canal.
- un canal est dit *non sûr* dans le cas contraire.

Un canal peut être sûr *physiquement* (exemple enveloppe scellée, téléphone rouge), soit par un procédé cryptographique. On a vu au passage la différence entre un adversaire *passif* — qui ne fait qu'écouter — et un adversaire *actif* — qui peut modifier les messages transmis, ou en émettre de nouveaux.

Un procédé cryptographique est dit *cassable* — ou *cassé* — s'il est possible, sans connaître les clés (d, e), de retrouver le texte clair correspondant à un texte chiffré donné, dans un certain temps.

Ce *certain temps* dépend de la donnée à protéger. Par exemple, le sujet d'examen du module d'introduction à la cryptologie n'a plus d'intérêt après janvier 2006, alors que la clé de commande de missiles nucléaires a une durée de vie plus grande.

Règle d'or 2. Il ne faut pas oublier de se prémunir contre l'attaque par recherche exhaustive sur l'espace $\mathcal{K}$ des clés. Cet ensemble doit donc contenir suffisamment d'éléments. Il est communément admis qu'une cardinalité de 2^{80}, soit environ 10^{24}, est suffisante. En effet, à raison d'un essai par cycle, sur une machine à 10Ghz, il faudrait environ 4 millions d'années pour parcourir un tel ensemble. À titre de comparaison, un espace de 2^{64} éléments pourrait être parcouru en 60 années seulement !

Lors de la conception d'un procédé devant être utilisé plusieurs années, il faut penser à la *loi de Moore* pour ne pas le sous-dimensionner. La loi de Moore indique en effet que la puissance des machines double environ tous les 18 mois. Exemple : si on veut un système pouvant résister 9 ans, il faut tenir compte du fait que la puissance des ordinateurs sera alors environ $2^6 = 64$ fois plus grande.