COURS ID12 « INTRODUCTION À LA CRYPTOLOGIE »
COURS 5
CHIFFREMENT ASYMÉTRIQUE : PANORAMA ET SIGNATURE

PAUL ZIMMERMANN

2. Chiffrement asymétrique

Remarque préliminaire : les algorithmes de chiffrement asymétrique sont en général plus lents que leurs homologues symétriques. On les emploie donc plutôt pour initialiser une session chiffrée (échange de clé de session symétrique) plutôt que pour chiffrer toute une session.

2.1. **L'échange de clé à la Diffie-Hellman.** Premier prototype d'un système à clé publique connu, avant que RSA ne soit découvert. Alice et Bob choisissent un groupe G, et un élément g de grand ordre n dans G. Alice choisit $a \in [1, n-1]$ et envoie à Bob $\alpha = g^a$; Bob choisit $b \in [1, n-1]$ et envoie à Alice $\beta = g^b$. Alice calcule β^a, et Bob calcule α^b. Alice et Bob partagent un secret commun $\beta^a = \alpha^b = g^{ab}$.

Pour trouver ce secret commun à partir de g^a et g^b, Charlie doit résoudre le problème du logarithme discret, i.e. retrouver a et b.

2.2. **Le système RSA.** C'est le premier système à clé publique, proposé en 1978 par Rivest, Shamir et Adleman.

2.2.1. *Génération de clé.* Alice choisit deux nombres premiers p et q, ainsi qu'un entier e aléatoire dans $[0, (p-1)(q-1)]$, premier avec $(p-1)(q-1)$. Elle calcule $N = p \cdot q$, et $d = e^{-1} \bmod (p-1)(q-1)$, par pgcd étendu. La *clé publique* est (N, e), la clé secrète est (p, q, d).

2.2.2. *Chiffrement.* Pour envoyer un message à Alice, Bob le coupe en morceaux qu'il peut encoder via des entiers inférieurs à N. Pour chaque morceau m, il calcule $c = m^e \bmod N$, qu'il envoie à Alice.

2.2.3. *Déchiffrement.* Pour déchiffrer c, Alice calcule $c^d \bmod N$. On a en effet, $c = m^e \bmod N$, donc $c^d = m^{ed} \bmod N$. On a aussi $c^d = m^{ed} \bmod p$ et $c^d = m^{ed} \bmod q$. Considérons $c^d \bmod p$: comme $ed = 1 \bmod (p-1)(q-1)$, on a $ed = 1 + \lambda(p-1)$ pour un certain entier λ. Donc $m^{ed} = m^{1+\lambda(p-1)}$. Or pour tout $m \neq 0$, $m^{p-1} = 1 \bmod p$ (petit théorème de Fermat), donc $m^{ed} = m \bmod p$, qui est vrai aussi pour $m = 0$. Le théorème des restes chinois permet de conclure que $m^{ed} = m \bmod N$.

2.2.4. *Coût.* Supposons que l'entier N a n bits. La génération de clé coûte deux recherches de nombres premiers d'environ $n/2$ bits, une multiplication et un inverse modulaire (pgcd étendu) sur n bits.

Le chiffrement coûte une exponentiation modulaire sur n bits, soit $O(nM(n))$. Le déchiffrement a le même coût : une exponentiation modulaire sur n bits, soit $O(nM(n))$.

Date: zimmerma@loria.fr.

2.2.5. *Attaques.* Il y a deux types d'attaques pour un système à clé publique :
- attaque sur la clé : étant donnée la clé publique, retrouver la clé secrète. Pour RSA, il s'agit, étant donné (N, e), de retrouver (p, q, d). On peut montrer que c'est équivalent à la factorisation de N.
- attaque sur le message : étant donné un message chiffré c, retrouver le message clair m correspondant. Pour RSA, comme $c = m^e \bmod N$, il s'agit d'extraire des racines e-ièmes modulo N.

2.2.6. *Sécurité.* Le record actuel de factorisation est de 200 chiffres décimaux (RSA-200), soit 663 bits. Cette factorisation a été annoncée par Bahr, Boehm, Franke et Kleinjung le 9 mai 2005. Il est recommandé d'utiliser une clé d'au moins 1024 bits. Aussi, p et q doivent être des nombres premiers forts, i.e. tels que $p-1$, $p+1$, $q-1$, $q+1$ ont un grand facteur premier. De même, si $r = (p-1)/2$ et $s = (q-1)/2$, $r-1$ et $s-1$ doivent avoir un grand facteur premier. L'exposant privé d ne doit pas être choisi trop petit ; par contre, on peut prendre e petit pour accélérer le chiffrement ($e = 65537$ est classique).

2.3. **Le système El Gamal.** Inventé par El Gamal en 1984. Son désavantage par rapport à RSA est que le message chiffré est deux fois plus gros que le clair.

2.3.1. *Génération de clé.* On choisit un groupe G, par exemple $(\mathbb{Z}_p)^*$, pour p premier bien choisi, et un élément g de grand ordre n dans G. Alice choisit a aléatoire dans $[2, n]$, et calcule $\gamma = g^a$. La clé publique est (G, g, γ), la clé secrète est a.

2.3.2. *Chiffrement.* Bob décompose son message en morceaux qu'il encode par des entiers inférieurs à $n = \mathrm{card}(G)$. Pour chiffrer un message clair m, il choisit b au hasard, calcule $z = g^b$, $w = \gamma^b$, et envoie $c = (z, wm)$.

2.3.3. *Déchiffrement.* Alice reçoit (u, v) et calcule $v/u^a = \gamma^b m/z^a = (g^a)^b m/(g^b)^a = m$.

2.3.4. *Exemple.* Soit $p = 2357$, $g = 2$, qui est d'ordre maximal 2356. Alice choisit $a = 1751$, et calcule $\gamma = 2^{1751} \bmod 2357 = 1185$. La clé publique est $(p = 2357, g = 2, g^a = 1185)$. Soit le message $m = 2035$; Bob choisit $b = 1520$, calcule $z = 2^{1520} \bmod 2357 = 1430$, et $w = 1185^{1520} \bmod 2357 = 2084$, puis $wm \bmod p = 697$. Bob envoie $(u = 1430, v = 697)$. Alice calcule $u^a = 1430^{1751} = 2084 \bmod p$, puis $v/u^a = 697/2084 = 2035 \bmod p$.

2.3.5. *Attaques.* L'attaque sur la clé consiste à déterminer a à partir de g et $\gamma = g^a$: c'est le problème du logarithme discret. L'attaque sur le message consiste à déterminer g^{ab} étant donnés g^a et g^b : c'est le problème dit de Diffie-Hellman.

2.3.6. *Complexité.* Pour p de n bits, la génération de clé coûte la recherche d'un nombre premier de n bits, plus une exponentiation modulaire. Le chiffrement coûte deux exponentiations et une multiplication modulaires, soit $O(nM(n))$. Le déchiffrement coûte une exponentiation et une division modulaires, soit aussi $O(nM(n))$.

2.3.7. *Sécurité.* Le record actuel est un calcul de logarithme discret dans $\mathrm{GF}(2^{613})$ (Joux, septembre 2005), soit 185 chiffres. Typiquement, le même type d'algorithme que pour la factorisation est mis en œuvre. On recommande donc des clés d'au moins 1024 bits également.

Références

1. Alfred J. Menezes, Paul C. van Oorschot, and Scott A. Vanstone, *Handbook of applied cryptography*, CRC Press, 1997, freely available at `http ://www.cacr.math.uwaterloo.ca/hac/`.