Attaques sur RSA

Ismail Baaj & Jade Tourteaux

2016

Table des matières

1 Introduction

Cryptosystème RSA RSA qui doit son nom à ses inventeurs Ron Rivest, Adi Shamir, et Len Adleman est le premier cryptosystème à clés publiques connu. Sa sécurité repose sur la difficulté à factoriser un entier positif qui est le produit de deux grands nombres premiers. Il repose sur l'utilisation d'une clé publique (qui est diffusée) et d'une clé privée (gardée secrète), l'une permettant de chiffrer un message et l'autre de le déchiffrer.

Création des clés Bob souhaite communiquer avec Alice. Elle doit obtenir une clé publique (N, e) et une clé privée d.

- Alice génère, au hasard et indépendamment deux nombres premiers p et q (arbitrairement grands), puis calcule le produit $N = pq$ ainsi que l'indicatrice d'euler de N, $\phi(N) = (p-1)(q-1)$.
- Elle choisit ensuite un entier e avec $1 < e < \phi(N)$ et $pgcd(e, \phi(N)) = 1$.
- Puis un autre entier d tel que $1 < d < \phi(N)$ tel que $de \equiv 1 \pmod{\phi(N)}$. Puisque e est premier avec $\phi(N)$, il est inversible modulo $\phi(N)$ donc le nombre d existe et on peut le calculer par l'algorithme d'Euclide étendu.

Elle possède donc (N, e) qui est sa *clé publique* et d sa *clé privée*.

On introduit aussi quelques définitions : le nombre N est appelé le *module*, e est *l'exposant de chiffrement*, et d *l'exposant de déchiffrement*.

Remarque : La clé privée peut être calculée à partir de l'exposant de chiffrement si les facteurs premiers p et q de N sont connus. Par conséquent, si une personne extérieure, Mathilde, est capable de trouver la décomposition en facteur premiers de N elle pourra facilement trouver la clé secrète d'Alice. La fiabilité de RSA repose donc sur la difficulté algorithmique du problème de *factorisation*

Chiffrement Un message en clair m est chiffré en calculant le cryptogramme $c \equiv m^e \pmod{N}$.

Déchiffrement Bob reçoit c, il va calculer $m \equiv c^d \pmod{N}$

Proposition. $(m^e)^d \equiv m \pmod{N}$

Preuve. Par définition $ed \equiv 1 \pmod{\phi(N)}$ donc $\exists k \in \mathbb{N}$ tel que
$ed = 1 + k\phi(N)$
D'où $(m^e)^d = m^{1+k\phi(N)} = m.(m^{\phi(N)})^k$. Or on a $m^{\phi(N)} \equiv 1 \pmod{N}$ d'après le Théorème d'Euler. On a donc bien $(m^e)^d \equiv m \pmod{N}$ □

2 Cryptanalyse Élémentaire

2.1 Méthode de Fermat

Tout entier naturel impair N se décompose en la différence de deux carrés : $N = a^2 - b^2$. On factorise cette différence en $N = (a+b)(a-b)$ et, si ni $a+b$ ni $a-b$ n'est égal à 1, alors ce sont des facteurs non triviaux de N.

Si on a le module RSA $N = pq$ avec p proche de q on peut factoriser rapidement N en cherchant les facteurs autour de $\sqrt{N}$. On utilise l'algorithme suivant :

Data: $N \in \mathbb{N}$
Result: p,q tels que $n = pq$
$a = \lfloor \sqrt{n} \rfloor$;
$b = \sqrt{a^2 - n}$;
while b *non entier* **do**
  $a = a + 1$;
  $b = \sqrt{a^2 - n}$;
end
$p = (a - b)$;
$q = (a + b)$;

Bien entendu, cette méthode se révèle inefficace dès que $\delta = |p - q|$ est trop grand, en effet, en conservant les notations de l'algorithme ci-dessus, $a = \frac{p+q}{2}$, le nombre d'opération est donc en $O(\frac{p+q}{2} - \sqrt{N})$ ce qui n'est pas raisonnable pour de grandes valeurs de N, p et q.

2.2 Module Commun

Soient deux clé publique (n, e_1) et (n, e_2), telles que e_1 premier avec e_2. Si un même message m est chiffré avec e_1 et e_2 alors un attaquant qui intercepterait les chiffrés pourrait les décrypter.

En effet, posons $c_1 = m^{e_1}$ et $c_2 = m^{e_2}$. Comme e_1 premier avec e_2, on a l'existence de u et v dans $\mathbb{Z}$ tels que $ue_1 + ve_2 = 1$ d'après le théorème de

Bézout. D'où $(c_1)^u(c_2)^v = m^{ue_1+ve_2} = m$.

Pour palier à ce problème, il faut intégrer de l'aléa dans les messages envoyés, par exemple avec du *padding*.

2.3 Exposant Commun

Pour cette attaque, on considère qu'un même message M est envoyé à k destinataires dont les clé publiques sont (N_i, e). Cette attaque dûe à Hastad est basée sur le théorème des restes chinois.

Proposition. *Soient les entiers N_i, $i \in \{1, \cdots, k\}$ deux à deux premiers entre eux. Alors le système*

$$\begin{cases} x & \equiv & a_1 & \pmod{N_1} \\ \vdots & \equiv & \vdots & \\ x & \equiv & a_k & \pmod{N_k} \end{cases}$$

admet pour unique solution

$$x \equiv \sum_{i=1}^{k} a_i p_i M_i \pmod{N}$$

avec $p_i = \frac{N}{N_i}$ et $M_i \equiv p_i^{-1} \pmod{N_i}$ et $N = \prod_{i=1}^{k} N_i$.

Preuve. Posons $N = \prod_{i=1}^{k} N_i$, $p_i = \frac{N}{N_i}$ et $M_i \equiv p_i^{-1} \pmod{N_i}$.
Alors pour chaque $i \in \{1, \cdots, k\}$ on a $p_i M_i \equiv 1 \pmod{N_i}$ et $N_i | p_j$ si $i \neq j$
Ainsi avec $x \equiv \sum_{i=1}^{k} a_i p_i M_i \pmod{N}$ on a $\forall i \in \{1, \cdots, k\}$

$$\begin{array}{rcll} x & \equiv & a_i p_i M_i + \sum_{j \neq i}^{k} a_j p_j M_j & \pmod{N_i} \\ x & \equiv & a_i + \sum_{j \neq i}^{k} a_j M_j N_i \times \frac{p_j}{N_i} & \pmod{N_i} \\ x & \equiv & a_i & \pmod{N_i} \end{array}$$

De plus cette solution est unique, en effet soit x' une autre solution $0 \leq x' < N$ alors $i \in \{1, \cdots, k\}$ on a

$$x' - x \equiv 0 \pmod{N_i}$$

Puisque les N_i sont premiers entre eux deux à deux

$$x' - x \equiv 0 \pmod{N}$$

Or $|x' - x| < N$ d'où $x' = x$ □

On en déduit une proposition adaptée à RSA

Proposition. *On pose $k \geq 2$, et les modules RSA N_i, $i \in \{1 \cdots k\}$ ainsi que le système d'équations $c_i \equiv m^e \pmod{N_i}$. Si $m^e < N = \prod_{i=1}^{k} N_i$, alors on peut retrouver m.*

Preuve. D'après la proposition précédente, on peut trouver x tel que $x \equiv c_i \pmod{N_i}$ avec $x < N$. Si $m^e < N$ alors $x = m^e$ d'où $m = x^{\frac{1}{e}}$. □

2.4 Indicatrice d'Euler

Si on connaît N et $\phi(N)$ remarquons que

$$\begin{aligned} & N - \phi(N) + 1 \\ &= pq - (p-1)(q-1) + 1 \\ &= pq - pq + p + q - 1 + 1 \\ &= p + q \end{aligned}$$

Posons $R(X) = X^2 - (N - \phi(N) + 1)X + n$

$$\begin{aligned} R(X) &= X^2 - (p+q)X + pq \\ R(X) &= (X-p)(X-q) \end{aligned}$$

On en déduit

$$p, q = \frac{(N - \phi(N) + 1) \pm \sqrt{(N - \phi(N) + 1)^2 - 4N}}{2}$$

2.5 Factoriser N avec l'exposant secret

Si on connaît N, e et d remarquons que $\forall a \in (\mathbb{Z}/N\mathbb{Z})^*$
$a^{de-1} = (a^{\phi(N)})^k \equiv 1 \pmod{N}$ avec k tel que $de - 1 = k\phi(N)$
On pose $r = de - 1$
r est forcément un nombre pair donc on peut écrire $r = 2^k m$, $2 \nmid m$.
Choisissons un $a \in (\mathbb{Z}/N\mathbb{Z})^*$. En calculant successivement les $a^m, \cdots, a^{2^i m}, \cdots, a^{2^k m}$, on a pour chaque $a^{2^i m}$

- Si $a^{2^i m} \equiv -1 \pmod{N} \Rightarrow$ changer de a
- Si $a^{2^i m} \equiv 1 \pmod{N} \Rightarrow pgcd(a^{2^i m} - 1, N)$ facteur non trivial de N
- Sinon on continue.

Proposition. *Soit $N = pq$ un module RSA, et d et e des exposants publiques et privés associés. On pose $r = de - 1 = 2^k m$ avec $2 \nmid m$. Alors il existe $a \in (\mathbb{Z}/N\mathbb{Z})^*$ et i un indice $1 \leq i \leq k$ tels que $pgcd(a^{2^i m} - 1, N)$ facteur non trivial de N. De plus*

$$\#\{a \in (\mathbb{Z}/N\mathbb{Z})^* \text{ tel que } \exists i \in \{0, \cdots, k-1\} \text{ et } 1 < pgcd(a^{2^i m} - 1, N) < N\} \geq \frac{\phi(N)}{4}$$

Preuve. L'algorithme ci-dessus est analogue au test de primalité de Miller-Rabin. On a $a^r \equiv 1 \pmod{N}$, comme $N = pq$ on a la système :

$$\begin{cases} a^r \equiv 1 \pmod{p} \\ a^r \equiv 1 \pmod{q} \end{cases}$$

Or r un nombre pair donc $a^r = (a^{2^{k-1} m})^2$ (en conservant les notations ci-dessus).

On a donc $(a^{2^{k-1}m})^2 \equiv 1 \pmod N$ d'où $a^{2^{k-1}m} \equiv \pm 1 \pmod N$.
On a donc à la i-ème étape de l'algorithme ;

$$(*) \begin{cases} a^{2^i m} & \equiv & \pm 1 \pmod p \\ a^{2^i m} & \equiv & \pm 1 \pmod q \end{cases}$$

Il y a donc quatre cas possibles pour le système $(*)$

1. $(+1,+1) \Rightarrow a^{2^i m}$ est encore un carré, on continue.
2. $(-1,-1) \Rightarrow$ cas où N passe le test de Miller-Rabin, donc on change de a.
3. $(+1,-1) \Rightarrow p = pgcd(a^{2^i m} - 1, N)$
4. $(-1,+1) \Rightarrow q = pgcd(a^{2^i m} - 1, N)$

Pour le cas (1), on a au maximum $\frac{\phi(N)}{2}$. Effet, comptons le nombre de carrés de $(\mathbb{Z}/N\mathbb{Z})^*$:
On utilise le morphisme de groupe $x \mapsto x^2$, dont le cardinal du noyau est égal à 2. Ainsi on le cardinal de l'image égal $\frac{\phi(N)}{2}$.

Pour le cas (2), le Théorème de Rabin nous donne une borne supérieure stricte en $\frac{\phi(N)}{4}$.

Ainsi on en déduit le nombre de a qui sont utiles pour l'algorithme

$$\#\{a \in (\mathbb{Z}/N\mathbb{Z})^* \text{ tel que } \exists i \in \{0, \cdots, k-1\} \text{ et } 1 < pgcd(a^{2^i m} - 1, N) < N\} \geq \frac{\phi(N)}{4}$$

□

3 Cryptanalyse de Wiener

Définition. *Soit $x \in \mathbb{R}$ alors x peut s'écrire sous la forme d'une fraction continue*

$$x = a_0 + \frac{1}{a_1 + \frac{1}{a_2 + \frac{1}{\dots}}}$$

On note $x = [a_0, a_1, a_2...]$

Définition. *Soit $x = [a_0, a_1, a_2...]$ une fraction continue d'un nombre x*

- *les nombres entiers a_i s'appellent les quotients partiels.*
- *les nombres rationnels $[a_0, a_1, a_2, ..., a_k]$ s'appellent les réduites de x.*

Théorème de Wiener Soit $N = pq$, avec $q < p < 2q$, et $d < \frac{1}{3}N^{\frac{1}{4}}$. Étant donnés N, e avec $ed = 1(mod(\phi(N)))$, l'attaquant peut facilement retrouver d. La complexité de cette attaque est $O(D^2)$ ou $d = Dn^{1/4}$.

Preuve du théorème de Wiener La preuve est basée sur l'approximation des fractions continues. On sait que $ed = 1 \pmod{()\phi(N)}$, il existe k tel que $ed - \phi(N) = 1$. Par conséquent :

$$\frac{e}{\phi(N)} - \frac{k}{d} = \frac{1}{d * \phi(N)}$$

D'où, $\frac{k}{d}$ est une approximation de $\frac{e}{\phi(N)}$, ainsi bien que l'attaquant ne connaisse pas $\phi(N)$, il va utiliser N pour l'approximer. En effet, étant donné que $\phi(N) = N - p - q + 1$ et $p + q - 1 < 3\sqrt{N}$, on a :

$$\begin{aligned} |p + q - 1| &< 3\sqrt{N} \\ |N + 1 - \phi(N) - 1| &< 3\sqrt{N} \end{aligned}$$

En utilisant N à la place de $\phi(N)$ on obtient :

$$\left|\frac{e}{N} - \frac{k}{d}\right| = \left|\frac{ed - kN}{Nd}\right| = \left|\frac{ed - k\phi(N) - kN + k\phi(N)}{Nd}\right| = \left|\frac{1 - k(N - \phi(N))}{Nd}\right|$$

On peut par la suite majorer ce résultat :

$$\left|\frac{1 - k(N - \phi(N))}{Nd}\right| \leq \left|\frac{3k\sqrt{N}}{Nd}\right| = \frac{3k\sqrt{N}}{\sqrt{N}\sqrt{N}d} = \frac{3k}{d\sqrt{N}}$$

Maintenant $k\phi(N) = ed - 1$ donc $k\phi(N) < ed$. On sait que $e < \phi(N)$, donc $k\phi(N) < ed < \phi(N)d$. On obtient $k\phi(N) < \phi(N)d$ et donc $k < d$.
Compte tenu de ce dernier résultat, et sachant que $d < \frac{1}{3}N^{\frac{1}{4}}$:

$$(1) \left| \frac{e}{N} - \frac{k}{d} \right| \leq \frac{1}{dN^{\frac{1}{4}}}$$

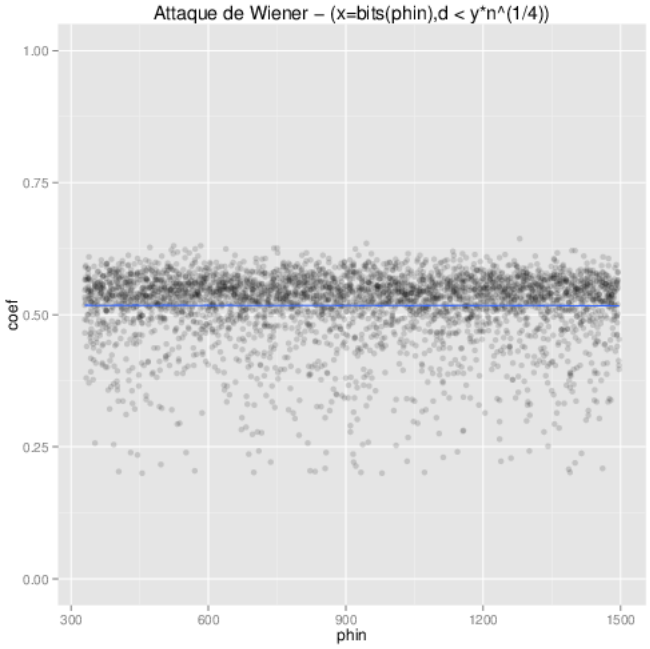
De $d < \frac{1}{3}N^{\frac{1}{4}}$ on en déduit que $ed < 3d$ donc $ed < 3d < N^{\frac{1}{4}}$ on obtient :

$$2d < N^{\frac{1}{4}} donc (2) \frac{1}{2d} > \frac{1}{N^{\frac{1}{4}}}$$

De (1) et (2) on peut en conclure que :

$$\left| \frac{e}{N} - \frac{k}{d} \right| \leq \frac{3k}{d\sqrt{N}} < \frac{1}{d * 2d} = \frac{1}{2d^2}$$

Borne de Wiener Nous avons effectué un travail sur la borne du théorème de Wiener. Initialement, le théorème permet de retrouver $d < \frac{1}{3}N^{\frac{1}{4}}$, et d'après un échantillon de 2000 clés RSA, on peut retrouver d pour $d < 0.52N^{\frac{1}{4}}$.



Attaque de de Weger En 2002, de Weger proposa une généralisation de l'attaque de Wiener. Cette attaque se base sur l'approximation de $\phi(n) \approx n + 1 - 2\sqrt{n}$. En utilisand $ed - k\phi(n) = 1$ et en supposant que $\phi(n) > \frac{3n}{4}$, $n > 8d$

avec $d < \frac{n^{\frac{3}{4}}}{|p-q|}$. Avec le même raisonnement que l'attaque de Wiener, on montre que

$$\left| \frac{e}{n+1-2\sqrt{n}} - \frac{k}{d} \right| < \frac{1}{2d^2}$$

Où $\frac{k}{d}$ est la convergence de fraction continues de $\frac{e}{n+1-2\sqrt{n}}$. Cette attaque repose sur la proximité de p et q.

4 Attaques de Coppersmith

4.1 Principe

Les attaques de Coppersmith sont basées sur le théorème de Coppersmith qui nous dit qu'on est capable de trouver les petites racines d'un polynôme modulaire en temps polynomial.

Théorème. *Soit N un entier dont on ne connaît pas la factorisation qui admet un diviseur b tel que $b \geq N^\beta$, avec $0 < \beta \leq 1$. On a $f(x)$ un polynôme de degré δ. Et une borne X. Alors on peut trouver tous les x_0 solutions de l'équation*

$$f(x) = 0 \pmod{b}$$

avec

$$|x_0| \leq X$$

en temps polynomial en δ et $\log N$.

L'idée, pour trouver ces solutions, est de passer du contexte de l'équation modulaire à une équation sur $\mathbb{Z}$. C'est-à-dire, de construire à partir de $f(x)$ un autre polynôme $g(x)$ qui admet pour racines les petites racines de $f(x)$. On cherche donc $g(x)$ tel que

$$f(x_0) = 0 \pmod{b} \Longrightarrow g(x_0) = 0 \text{ sur } \mathbb{Z},\ \forall |x_0| \leq X$$

En effet, avoir un tel polynôme nous donne directement les x_0 recherchés car il suffit de factoriser le polynôme sur $\mathbb{Z}$ grâce aux méthodes courantes [1]. Pour trouver un tel g, Coppersmith propose deux étapes

1. On fixe un entier m, et on construit une famille de polynômes f_i : $f_1(x), f_2(x), \cdots, f_n(x)$ qui admettent tous pour racines les racines x_0 de f telles que $|x_0| \leq X$.
2. On construit une combinaison linéaire $g(x) = \sum_{1=1}^{n} a_i f_i(x)$, $a_i \in \mathbb{Z}$ telle que $|g(x_0)| < b^m$. On remarque que $b^m | f_i(x_0)$ par construction, donc que $b^m | g(x_0)$. Mais comme $g(x_0) = 0 \pmod{b^m}$ et $|g(x_0)| < b^m$ on a donc $g(x) = 0$ sur $\mathbb{Z}$.

La construction (2) sera faite dans un contexte de réseau, et on se servira de l'algorithme LLL pour trouver le polynôme g correspondant.

1. Algorithme de Berlekamp-Zassenhaus, ou encore LLL :-)

4.2 LLL

Définition. *(Réseau) Soit n un entier positif. Un sous-ensemble L de $\mathbb{R}^n$ est un réseau s'il existe des vecteurs $b_1, b_2, \cdots b_n \in \mathbb{R}^n$ tels que*

$$L = \sum_{i=1}^{n} \mathbb{Z} b_i = \left\{ \sum_{i=1}^{n} r_i b_i | r_i \in \mathbb{Z} (1 \leq i \leq n) \right\}$$

On dit que les b_i forment une base de L, on appelle n le rang de L et le déterminant de L

$$det(L) = |det(b_1, b_2, \cdots, b_n)|$$

L'algorithme LLL (du nom de ses créateurs : Lenstra, Lenstra, Lovász) prend en entrée une base quelconque d'un réseau, et retourne une base de réseau réduite, c'est-à-dire presque orthogonale, composée de vecteurs courts.

Définition. *(Base Réduite) Soient $b_1, b_2, \cdots, b_n$ une base. On utilise le procédé d'orthogonalisation de Gram-Schmidt. On définit ainsi les vecteurs b_i^*, pour $1 \leq i \leq n$ et les nombres réels $\mu_{i,j}$, pour $1 \leq j < i \leq n$*

$$b_i^* = b_i - \sum_{j=1}^{i-1} \mu_{i,j} b_j^*$$

$$\mu_{i,j} = \frac{(b_i, b_j^*)}{(b_j^*, b_j^*)}$$

où $(.,.)$ est le produit ordinaire sur $\mathbb{R}^n$.

On appelle base réduite de L une base telle que

1. *$|\mu_{i,j}| \leq \frac{1}{2}$, pour $1 \leq j < i \leq n$*
2. *$|b_i^j + \mu_{i,i-1} b_{i-1}^*|^2 \geq \frac{3}{4} |b_{i-1}^*|^2$, pour $1 < i \leq n$*

où $|.,.|$ est une norme sur $\mathbb{R}^n$

Proposition. *Soient $b_1, b_2, \cdots, b_n$ une base réduite pour une réseau L de $\mathbb{R}^n$ et les $b_1^*, \cdots, b_n^*$ définis comme précédemment. Alors*

1. *$|b_j|^2 \leq 2^{i-1} |b_i^*|^2$, pour $1 \leq j \leq i \leq n$*
2. *$det(L) \leq \prod_{i=1}^{n} |b_i| \leq 2^{\frac{n(n-1)}{4}} det(L)$*
3. *$|b_1| \leq 2^{\frac{n-1}{4}} det(L)^{\frac{1}{n}}$*

Preuve. De la définition on a $|b_j|^2 \geq (\frac{3}{4} - \mu_{i,i-1}^2) |b_{i-1}^*|^2 \geq \frac{1}{2} |b_{i-1}^*|^2$ pour $1 < i \leq n$. D'où

$$|b_j^*|^2 \leq 2^{i-j} |b_i^*|^2$$

pour $1 \leq j \leq i \leq n$

$$\begin{aligned} |b_i|^2 &= |b_i^*|^2 + \sum_{j=1}^{i-1} \mu_{i,j}^2 |b_j^*|^2 \\ &\leq |b_i^*|^2 + \sum_{j=1}^{i-1} \tfrac{1}{4} 2^{i-j} |b_i^*|^2 \\ &= (1 + \tfrac{1}{4}(2^i - 2)|b_i^*|^2 \\ &\leq 2^{i-1}|b_i^*|^2 \end{aligned}$$

Il s'ensuit que

$$|b_i|^2 \leq 2^{j-1}|b_j^*|^2 \leq 2^{i-1}|b_i^*|^2$$

pour $1 \leq j \leq i \leq n$ On a $det(L) = |det(b_1^*, \cdots, b_n^*)|$ car les b_i^* forment une base, et comme ils sont deux à deux orthogonaux

$$det(L) = \prod_{i=1}^{n} |b_i^*|$$

Comme $|b_i^*| \leq |b_i|$ et $|b_i \leq 2^{\frac{i-1}{2}}|b_i^*|$ on obtient

$$det(L) \leq \prod_{i=1}^{n} |b_i| \leq 2^{\frac{n(n-1)}{4}} det(L)$$

Enfin, si on prends $j = 1$ dans $|b_j|^2 \leq 2^{i-1}|b_i^*|^2$, en utilisant (2) on trouve $|b_1| \leq 2^{\frac{n-1}{4}} det(L)^{\frac{1}{n}}$ □

On a là un résultat important pour la suite, l'existence d'un vecteur v dans la base réduit par LLL tel que

$$|v| \leq 2^{\frac{n-1}{4}} det(L)^{\frac{1}{n}}$$

4.3 Résolution d'une équation polynomiale modulaire

Coppersmith énonce en 1996 son théorème dont la démonstration nous donne une manière effective pour résoudre une équation polynomiale modulaire.

Théorème (Coppersmith 1)**.** *Soit N un entier dont on ne connaît pas la factorisation et $f_N(x)$ un polynôme de degré δ. On peut trouver tous les x_0 solutions de l'équation*

$$f_N(x) = 0 \pmod{N}$$

avec

$$|x_0| \leq N^{\frac{1}{\delta}}$$

en temps polynomial en δ et $\log N$.

Ce théorème est celui utilisé en pratique dans les attaques de RSA, mais c'est un cas particulier du théorème suivant :

Théorème (Coppersmith 2)**.** *Soit N un entier dont on ne connaît pas la factorisation qui admet un diviseur b tel que $b \geq N^\beta$, avec $0 < \beta \leq 1$.*

Soit ϵ, $0 < \epsilon \leq \frac{1}{7}\beta$. Et $f(x)$ un polynôme univarié de degré δ. Alors on peut trouver toutes les solutions x_0 de l'équation

$$f(x) = 0 \pmod{b}$$

avec

$$|x_0| \leq \frac{1}{2} N^{\frac{\beta^2}{\delta} - \epsilon}$$

en temps polynomial en δ et $\log N$.

Pour démontrer ce théorème, on utilise le théorème d'Howgrave-Graham qui nous donne les conditions nécessaires pour passer d'une équation modulaire à une équation dans $\mathbb{Z}$.

Théorème (Howgrave-Graham)**.** *Soit $g(x) \in \mathbb{Z}[x]$ un polynôme de degré d ayant au plus n monômes et $X \in \mathbb{N}$. Si x_0 un entier et m un nombre positif tel que*

1. $|x_0| < X$
2. $g(x_0) \equiv 0 \pmod{b^m}$
3. $||g(xX)|| < \frac{b^m}{\sqrt{n}}$

Alors $g(x_0) = 0$ dans $\mathbb{Z}$

Preuve. Soit $g(x) = \sum_i^d a_i x^i$ ayant n monômes et $x_0 < X$

$$\begin{array}{rcl} |g(x_0)| & = & \left|\sum a_i x^i\right| \\ \left|\sum a_i x^i\right| & < & \sum |a_i x^i| \\ \sum |a_i x^i| & < & \sum |a_i X^i| \end{array}$$

Or d'après l'inégalité de Cauchy-Schwarz

$$\begin{array}{rcl} (\sum |a_i X^i|)^2 & \leq & (\sum 1^2)(\sum (a_i X^i)^2) \\ (\sum 1^2)(\sum (a_i X^i)^2) & = & n \sum (a_i X^i)^2 \end{array}$$

De plus si $||g(xX)|| < \frac{b^m}{\sqrt{n}}$

$$\sum |a_i X^i| < \sqrt{n} \sqrt{\sum (a_i X^i)^2} = \sqrt{n} ||g(xX)|| < b^m$$

donc $|g(x_0)| < b^m$ □

Si on reprend les deux étapes de données en 4.1, on a en (1) construit une famille de polynômes $f_1(x), f_2(x), \cdots, f_n(x)$ qui ont les racines x_0 désirées modulo b^m , et avec (2) on obtient $g(x)$ tel que $g(x_0) = \sum_{1=1}^n a_i f_i(x_0) = 0 \pmod{b^m}$,

$a_i \in \mathbb{Z}$. Donc toutes les combinaisons linéaires satisfont la condition (1) du théorème d'Howgrave-Graham.

On en cherche donc une qui satisfait la condition (2), c'est-à-dire on cherche dans les combinaisons linéaires des $f_i(xX)$ un vecteur dont la norme est inférieure à $\frac{M}{\sqrt{w}}$. On peut trouver un tel vecteur grâce à l'algorithme LLL. En effet on sait qu'il existe dans la base LLL-réduite de la matrice L représentant une réseau de dimension n un vecteur v tel que

$$||v|| \leq 2^{\frac{n-1}{4}} det(L)^{\frac{1}{n}}$$

Ainsi, si on avait $2^{\frac{n-1}{4}} det(L)^{\frac{1}{n}} < \frac{N^{\beta m}}{\sqrt{n}}$ alors on aurait $||v|| < \frac{N^{\beta m}}{\sqrt{n}} \leq \frac{b^m}{\sqrt{n}}$.

Si on néglige les termes qui ne dépendent pas de N, on obtient la condition simplifiée

$$det(L) < N^{\beta mn}$$

Soit L un réseau de dimension n qui satisfait cette condition, alors un vecteur v de la base, contribue en moyenne au déterminant avec un facteur inférieur à $N^{\beta} m$.

Nous avons tous les éléments pour démontrer le théorème de Coppersmith 2.

Théorème (Coppersmith 2)**.** *Soit N un entier dont on ne connaît pas la factorisation qui admet un diviseur b tel que $b \geq N^{\beta}$, avec $0 < \beta \leq 1$.*

Soit ϵ, $0 < \epsilon \leq \frac{1}{7}\beta$. Et $f(x)$ un polynôme univarié de degré δ. Alors on peut trouver toutes les solutions x_0 de l'équation

$$f(x) = 0 \pmod{b}$$

avec

$$|x_0| \leq \frac{1}{2} N^{\frac{\beta^2}{\delta} - \epsilon}$$

La complexité est dominée par la complexité de la réduction LLL d'une base d'un réseau de dimension en $O(\epsilon^{-1}\delta)$ avec des entrées de taille en $O(\epsilon^{-1} \log N)$. Cela peut être fait en temps en $O(\epsilon^{-7}\delta^5 \log^2 N)$.

Démonstration. Posons $X = \frac{1}{2} N^{\frac{\beta^2}{\delta} - \epsilon}$ et appliquons les deux étapes décrites en 4.1.

1. On fixe $m = \left\lceil \frac{\beta^2}{\delta\epsilon} \right\rceil$. On prends des polynômes qui ont pour racine $x_0 \pmod{b^m}$ quand $f(x)$ a une racine modulo b.

   $$g_{i,j}(x) = x^j N^i f^{m-i}(x)$$

   où $j = 0, \cdots, \delta - 1$ et $i = 0, \cdots, m - 1$ et les

   $$h_i(x) = x^i f^m(x)$$

   où $i = 0, \cdots, t - 1$, t étant un entier fixé, dépendant de m.

2. On construit la matrice L de taille $n = m\delta + t$ définie par les coefficients des $g_{i,j}$ et h_i ligne par ligne dans la base $(1, x, x^2, \cdots, x^{m\delta+t-1})$.

$$\left(\begin{array}{c} g_{m,0} \\ g_{m,1} \\ g_{m,2} \\ \vdots \\ g_{m,d-1} \\ \hline g_{m-1,0} \\ g_{m-1,1} \\ \vdots \\ g_{m-1,d-1} \\ \hline \vdots \\ \hline g_{1,0} \\ \vdots \\ g_{1,d-1} \\ \hline h_0 \\ \vdots \\ h_{t-1} \end{array}\right)$$

Comme les degrés de ces polynômes sont croissants de la position$(i, j) = (m, 0)$ à $(i, j) = (1, \delta - 1)$ et de même pour les polynômes h_i on a alors une matrice triangulaire inférieure.
Son déterminant est donc le produit de ses éléments sur la diagonale. En explicitant les polynômes on obtient

$$\begin{aligned} det(L) &= N^{m\delta} * N^{(m-1)\delta} * \cdots * N^{\delta} X^{1+2+\cdots+n-1} \\ det(L) &= N^{\frac{1}{2}m(m+1)\delta} X^{\frac{1}{2}n(n-1)} \end{aligned}$$

Comment choisir le paramètre t ? Optimiser le choix de t c'est optimiser $n = \delta m + t$. On sait que les coefficients des vecteurs sur la diagonale de la matrice doivent être inférieurs à $N^{\beta m}$ (d'après la remarque précédent l'énoncé du théorème), ce qui nous donne une condition sur X :

$$X^{n-1} < N^{\beta m}$$

Comme on a

$$X^{n-1} < N^{(\frac{\beta^2}{\delta} - \epsilon)(n-1)} < N^{(\frac{\beta^2}{\delta} n}$$

On obtient la condition

$$n \leq \frac{\delta}{\beta} m$$

D'après notre manière de fixer m on sait que $m \leq \frac{\beta^2}{\delta\epsilon} + 1$. Donc on a une borne pour la dimension de la matrice $n \leq \frac{\beta}{\epsilon} + \frac{\delta}{\beta}$.

En utilisant $7\beta^{-1} \leq \epsilon^{-1}$ on obtient $n = O(\epsilon^{-1}\delta)$. On choisit pour n le maximum qui satisfait $n \leq \frac{\delta}{\beta}m$. On a donc $n > \frac{\delta}{\beta}m - 1 \geq \frac{\beta}{\epsilon} - 1 \geq 6$

Cherchons maintenant la taille des entrées de la base. Pour toutes les puissances f^{m-i} de la définition des $g_{i,j}$ et h_i on peut réduire les coefficients modulo N^{m-i} car les x_0 doivent être racines modulo N^{m-i}. Ainsi le coefficient le plus grand dans le produit des $N^i f^{m-i}$ est de taille maximum $m \log N = O(\epsilon^{-1} \log N)$.

Les puissances de $X = \frac{1}{2}N^{\frac{\beta^2}{\delta}-\epsilon}$ apparaissent avec un exposant inférieur à n donc la taille des puissances de X peut être majorée par $\frac{\beta^2}{\delta} \log N = O(\frac{\delta}{\epsilon}.\frac{\beta^2}{\delta}) \log N = O(\epsilon^{-1} \log N)$.

La complexité de LLL est centrale pour le temps d'exécution. Si on considère la version LLL de Nguyen et Stehlé qui propose la même borne pour le vecteur court mais un temps d'exécution en $O(n^5(n + \log b) \log b)$ [2], où $\log b$ est la taille maximale d'une entrée de la base à réduire. On obtient donc une complexité en $O((\frac{\delta}{\epsilon})^5(\frac{\delta}{\epsilon} + \frac{\log N}{\epsilon})\frac{\log N}{\epsilon})$. En pratique on peut facilement supposer que $\delta \leq \log N$. On a donc un temps d'exécution effectif en $O(\epsilon^{-7}\delta^5 \log^2 N)$.

Il reste à montrer que l'approximation du court vecteur de LLL est suffisant pour le théorème d'Howgrave-Graham, c'est à dire un vecteur dont la norme est inférieure à $\frac{b^m}{\sqrt{n}}$.

On sait que $||v|| \leq 2^{\frac{n-1}{4}} det(L)^{\frac{1}{n}}$. On doit montrer que $2^{\frac{n-1}{4}} det(L)^{\frac{1}{n}} < \frac{b^m}{\sqrt{n}}$. On a le déterminant de la matrice est $det(L) = N^{\frac{1}{2}m(m+1)\delta} X^{\frac{1}{2}n(n-1)}$ et $b \geq N^{\beta}$. On a donc $N^{\frac{\delta m(m+1}{2}} X^{\frac{n(n-1)}{2}} \leq 2^{-\frac{n-1}{4}} n^{-\frac{1}{2}} N^{\beta m}$. Cela nous donne la condition sur la taille de la borne X

$$X \leq 2^{-\frac{1}{2}} n^{-\frac{1}{n-1}} N^{\frac{2\beta m}{n-1} - \frac{\delta m(m+1)}{n(n-1)}}$$

On remarque que $n^{-\frac{1}{n-1}} = n^{-\frac{\log n}{n-1}} \geq 2^{-\frac{1}{2}}$ pour $n \geq 6$. La condition se simplifie donc en

$$X \leq \frac{1}{2} N^{\frac{2\beta m}{n-1} - \frac{\delta m(m+1)}{n(n-1)}}$$

Or on a choisi $X = \frac{1}{2}N^{\frac{\beta^2}{\delta}-\epsilon}$, il suffit donc de montrer que

$$\frac{2\beta m}{n-1} - \frac{\delta m^2(1 + \frac{1}{m})}{n(n-1)} \geq \frac{\beta^2}{\delta} - \epsilon$$

On utilise le fait que $n \leq \frac{\delta}{\beta}m$, ainsi on obtient

$$2\frac{\beta^2}{\delta} - \frac{\beta^2}{\delta}(1 + \frac{1}{m}) \geq \frac{\beta^2}{\delta} - \epsilon$$

2. Phong Nguyen and Damien Stehlé, "Floating-point LLL Revisited"

Ce qui se simplifie en

$$-\frac{\beta^2}{\delta}.\frac{1}{m} \geq -\epsilon$$

Ce qui nous donne la condition sur m, $m \geq \frac{\beta^2}{\delta\epsilon}$ ce qui correspond bien au choix de m du début. □

On déduit de cette démonstration, un algorithme pour la méthode de Coppersmith.

Entrée : Un polynôme $f(x)$ de degré δ, un module N de factorisation inconnue qui admet une diviseur b tel que $b \geq N^\beta, \epsilon \leq \frac{1}{7}\beta$.

1. On calcule $m = \left\lceil \frac{\beta^2}{\delta\epsilon} \right\rceil$ et $t = \lfloor \delta m(\frac{1}{\beta} - 1) \rfloor$
   $g_{i,j}(x) = x^j N^i f^{m-i}(x)$ où $j = 0, \cdots, \delta - 1$ et $i = 0, \cdots, m-1$
   $h_i(x) = x^i f^m(x)$ où $i = 0, \cdots, t-1$
2. $X = \frac{1}{2}N^{\frac{\beta^2}{\delta} - \epsilon}$. On construit le réseau dont une base est formée des vecteurs coefficients des $g_{i,j}(x)$ et $h_i(x)$
3. On applique LLL sur la base. Soit v le plus court vecteur de la base, on construit $g(x)$ à partir de v.
4. On trouve l'ensemble R des racines de $g(x)$ sur $\mathbb{Z}$ et $\forall x_0 \in R$ on vérifie si $pgcd(N, f(x_0)) \geq N^\beta$ si la condition n'est pas remplie, retirer x_0 de R.

Sortie : Ensemble R où $x_0 \in R \Leftrightarrow f(x_0) = 0 \pmod b$ et $|x_0|$.

Note : on effectue une vérifications en 4. car $g(x)$ peut contenir des racines qui ne sont pas des racines de $f(x) \pmod b$ donc on teste si $f(x_0)$ contient un diviseur de N de taille au moins N^β .

On peut éviter le facteur $\frac{1}{2}$ et l'utilisation du ϵ dans la borne pour les x_0

Théorème. *Soit N un entier dont on ne connaît pas la factorisation qui admet un diviseur b tel que $b \geq N^\beta$, avec $0 < \beta \leq 1$. Et $f(x)$ un polynôme univarié de degré δ. Alors on peut trouver toutes les solutions x_0 de l'équation*

$$f(x) = 0 \pmod b \text{ avec } |x_0| \leq cN^{\frac{\beta^2}{\delta}}$$

en temps en $O(c\delta^5 \log^9 N)$.

Démonstration. On applique le Théorème 2 avec $\epsilon = \frac{1}{\log N}$, donc on peut trouver les x_0 vérfiants

$$|x_0| \leq \frac{1}{4}N^{\frac{\beta^2}{\delta}}$$

en temps en $O(\delta^5 \log^9 N)$ Pour trouver toutes les racines qui sont de taille maximum $cN^{\frac{\beta^2}{\delta}}$ en valeur absolue on divise l'intervalle $[-cN^{\frac{\beta^2}{\delta}}, cN^{\frac{\beta^2}{\delta}}]$ en $4c$ sous-intervalles de taille $\frac{1}{2}N^{\frac{\beta^2}{\delta}}$ centrés autour d'un certain x_i. Pour chaque sous-intervalle on applique le théorème 2 au polynôme $f(x - x_i)$. On trouve ainsi toutes les solutions. □

On en déduit le théorème de Coppersmith qui s'applique à RSA en prenant le cas particulier $b = N$ et $c = 1$

Théorème (Coppersmith 1)**.** *Soit N un entier dont on ne connaît pas la factorisation et $f_N(x)$ un polynôme de degré δ. On peut trouver tous les x_0 solutions de l'équation*

$$f_N(x) = 0 \pmod{N} \text{ avec } |x_0| \leq N^{\frac{1}{\delta}}$$

en temps en $O(\delta^5 \log^9 N)$.

4.4 Application de la méthode de Coppersmith

4.4.1 Message stéréotypé

Si l'attaquant connaît le début du message (typiquement un message "la clef du jour est ...") alors il peut décrypter le message en entier.
Supposons que l'on ai un chiffré c pour un message $m = B + x$, tel que $B = 2^k b$ connu et x inconnu. De plus on connaît la clef publique (n, e) qui a servi à chiffrer m.
Alors $c = (B+x)^e \pmod{n}$, ce qui nous donne l'équation polynomiale modulaire

$$(B + x)^e - c = 0 \pmod{n}$$

En utilisant le théorème de Coppersmith, on peut résoudre cette équation.

4.4.2 Attaque petit exposant privé

L'attaque de Boneh-Durfee nous permet de casser RSA et la clé privée d. On sait que $ed = 1 \pmod{\phi(N)}$.

$$\begin{aligned} ed &= k\phi(N) + 1 \text{ sur } Z \\ k\phi(N) + 1 &= 0 \pmod{e} \\ k(N + 1 - p - q) + 1 &= 0 \pmod{e} \\ 2k[(N + 1)/2 + (-p - q)/2] + 1 &= 0 \pmod{e} \end{aligned}$$

La dernière équation nous donne un polynôme bivarié $f(x, y) = 1 + x * (A + y)$. Si on trouve les racines de ce polynôme cela nous permet de facilement calculer la clé privée d.

Cette attaque fonctionne pour $d < N^{0.292}$

4.4.3 Connaissance des LSB

En pratique RSA peut être embarqué dans une carte à puce, tant pour le stockage de la clé privée que pour l'algorithme de déchiffrement. Il est possible pour un attaquant d'obtenir des informations par *side channel attack* (attaque par canaux cachés) par exemple en analysant le temps de calcul ou la chaleur dégagée par la carte.

Il est donc raisonnable de considérer des attaques basées sur la connaissance d'une fraction de bits consécutifs de la clef privée.

Proposition. *Soit $N = pq$ un module RSA de n bits. Si on connaît $\frac{n}{4}$ bits de poids faible de la clé privée d alors, on peut reconstituer d en temps polynomial en n et e.*

Preuve. Admettons que l'on connaisse $\frac{n}{4}$ des bits de poids faible de d. Par définition de e et d, il existe une entier k tel que

$$ed - k(N - p - q + 1) = 1$$

Comme $d < \phi(N)$ on doit avoir $0 \leq k \leq e$. Si on pose $q = \frac{N}{p}$ et que l'on réduit l'équation modulo $2^{\frac{n}{4}}$ on obtient

$$(ed)p - kp(N - p + 1) + kN = p \pmod{2^{\frac{n}{4}}}$$

Or on connaît $\frac{n}{4}$ bits de poids faible de d donc on connaît la valeur de $ed \pmod{2^{\frac{n}{4}}}$. Ainsi nous avons obtenu une équation en p et k.

Or $0 < k \leq e$, donc pour les e valeurs possibles de k on résout l'équation en p modulo $2^{\frac{n}{4}}$ grâce à la méthode de Coppersmith. □

Comme l'exposant publique e est souvent petit sur une carte à puce pour accélérer les phases de chiffrement/déchiffrement, cette attaque est efficace en pratique. Elle a été présentée par Boneh, Durfee et Frankel en 1998.

5 OpenSSL

Développé en C et étant sous licence libre, OpenSSL est un logiciel de chiffrement utilisé par environ deux tiers des sites-web. Il possède une implémentions du cryptosystème RSA, que nous avons étudié.

Caractéristiques de l'implémentation de RSA OpenSSL impose que e doit être soit 3 ou 65537 (le plus grand nombre premier de Fermat) - d sera donc très grand est proche de n.
Après génération, OpenSSL effectue quelques vérifications rudimentaires du cryptosystème RSA que l'on retrouve dans le fichier *rsa_chk.c*, permettant d'assurer que celle-ci respecte bien les données de l'algorithme.
On obtient les clés générées dans des fichiers au format DER ou PEM (ce dernier est le chiffrement base64 de la version DER). On note que la clé privée peut être chiffrée à l'aide de AES, DES ou encore DES3 par exemple.

Structure des clés : exemple avec un couple de clés RSA 256bits

clé publique

Format PEM

```
-----BEGIN PUBLIC KEY-----
MDwwDQYJKoZIhvcNAQEBBQ
ADKwAwKAIhAModa9Vucw4
aKfa+gQ7SZilA9bo3Go8l
W2MjEAzKIAXPAgMBAAE=
-----END PUBLIC KEY-----
```

Convertie en hexadecimal

```
-----BEGIN PUBLIC KEY-----
303C300D06092A864886F70D01
01010500032B0030280221 00CA1
D6BD56E730E1A29F6BE810ED
2662940F5BA371A8F255B63231
00CCA2005CF 0203 010001
-----END PUBLIC KEY-----
```

Les éléments qui composent la clé sont les suivants :
- 303C3 ... 0030 : l'en tête (*header*)
- 00CA ... 05CF : N, le module
- 010001 : e, l'exposant publique (0x10001 étant égal à 65537)
- les autres éléments sont les séparateurs.

clé privée

Format PEM

```
-----BEGIN RSA PRIVATE KEY-----
MIGpAgEAAiEAyh1r1W5zDhop
9r6BDtJmKUD1ujcajyVbYyMQD
MogBc8CAwEAAQIgewRXSuvT6a
RxIDapdK0I8kJGStZRJLue
YXB3J5yDXUECEQD/pU6Amg
KfC7rky4Jsn1AvAhEAym
UftziXlYVItxKg8fqcYQIRAJ
Pi9P7AGTQYfQ1ODSz5HFECE
HKA1XmdzbxspF/WbTGZG+EC
D2UZTlrldnkbOktnHtl2Hw==
-----END RSA PRIVATE KEY-----
```

Hexadecimal

```
-----BEGIN RSA PRIVATE KEY-----
3081A9 02 01 00 02 21 00CA1D6BD
56E730E1A29F6BE810ED26629
40F5BA371A8F255B6323100CC
A2005CF 02 03 010001 02 20 7B0457
4AEBD3E9A4712036A974AD08
F242464AD65124BB9E61707727
9C835D41 02 11 00FFA54E809A0
29F0BBAE4CB826C9F502F 02 1
100CA651FB73897958548B712A
0F1FA9C61 02 11 0093E2F4FEC0
1934187D0D4E0D2CF91C51 02 1
07280D5799DCDBC6CA45FD66
D31991BE1 02 0F 65194E5AE5767
91B3A4B671ED9761F
-----END RSA PRIVATE KEY-----
```

Les éléments qui composent la clé sont les suivants :

- 3081A9 : l'en tête (*header*)
- 00 : la version de l'algorithme
- 00CA1D6BD ... A2005CF : N, le module
- 010001 : e, l'exposant publique (0x10001 étant égal à 65537)
- 7B0457 ... 9C835D41 : d, l'expostant privé
- 00FF ... 9F502F : p
- 00CA ... 9C61 : q
- 0093 ... 1C51 : $d \pmod{(p-1)}$
- 7280 ... 1BE1 : $d \pmod{(q-1)}$
- 6519 ... 761F : $q^{-1} \pmod{p}$
- les autres éléments sont des séparateurs

Attaque Du fait de la grande proximité de d et n, une méthode efficace est de chercher une factorisation de n à l'aide de l'algorithme $Msieve$.

Résultats Pour une clé de 256bits, nous réussissons à trouver une factorisation en environ 2minutes avec un ordinateur doté d'un processeur $i7$. Elle est immédiate pour une clé de 64bits.
Il nous est alors possible de reconstruire un fichier de clé privée au format PEM.

Limites $Msieve$ est efficace pour n de taille inférieur à 110 chiffres (il faut compter environ 120 heures avec un processeur moderne pour obtenir une factorisation d'un nombre de cette taille). Aujourd'hui l'algorithme le plus efficace de factorisation est le crible général de corps de nombres, appelé aussi crible algébrique, qui demande $O\left\{\exp\left[\left(\frac{64}{9}\log n\right)^{\frac{1}{3}}(\log\log n)^{\frac{2}{3}}\right]\right\}$ étapes pour factoriser un nombre entier n.

Records De nombreux records de factorisation sont souvent établis. La société $RSA-Security$ organisait même une compétition de factorisation jusqu'en 2007. Le record actuel est une factorisation d'un nombre de 768 bits (appelé $RSA-768$) factorisé le 12 décembre 2009 par une équipe de 13 chercheurs à l'aide d'un ensemble d'ordinateurs effectuant des calculs en parallèle. Le temps nécessaire pour trouver ces facteurs correspond à 2000 années de calcul d'un simple *core* d'un processeur AMD Opteron 2.2Ghz.
Dans la pratique, aujourd'hui, il est recommandé de faire usage de clés de taille 2048 bits.