ATTAQUES SUR RSA

Ismail Baaj & Jade Tourteaux

2015

Université Paris Diderot

CRYPTOSYSTÈME RSA

Le cryptosystème RSA est un chiffrement asymétrique :

- (N, e) est la clé publique
- d est la clé privée

tel que

- $N = pq$ (avec p et q très grand nombres premiers)
- $pgcd(e, \phi(N)) = 1$
- $ed \equiv 1 \pmod{\phi(N)}$

Chiffrement : $c \equiv m^e \pmod{N}$
Déchiffrement : $m \equiv c^d \pmod{N}$

MÉTHODE DE FERMAT

On factorise N, en utilisant le fait que tout nombre impair est la différence de deux carrés : $N = a^2 - b^2$. D'où la factorisation

$$N = (a+b)(a-b)$$

Algorithm 1 Factoriser N

```
1: a = ⌊√N⌋
   b = √(a² − N)
2: while b non entier do
3:     a = a + 1
       b = √(a² − N)
4: end while
5: return p = (a − b) et q = (a + b)
```

Cette méthode est pertinente dans le cas où p serait proche de q et donc de la racine de N.

- Deux clés publiques (n, e_1) et (n, e_2), e_1 premier avec e_2
- Un message m chiffré avec e_1 et e_2
- $c_1 = m^{e_1}$ et $c_2 = m^{e_2}$

Comme e_1 premier avec e_2, on a d'après le théorème de Bézout deux entiers u et v tels que $ue_1 + ve_2 = 1$, on les trouve grâce à l'algorithme d'Euclide Étendu.

$$(c_1)^u(c_2)^v = m^{ue_1+ve_2} = m$$

Pour éviter cette attaque, il faut intégrer de l'aléa dans les messages envoyés.

Cette attaque est due à Hastad. On utilise la proposition basée sur le théorème chinois

Proposition

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

$$\left\{ \begin{array}{ccll} x & \equiv & a_1 & \pmod{N_1} \\ \vdots & \equiv & \vdots & \\ x & \equiv & a_k & \pmod{N_k} \end{array} \right.$$

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 $N = \prod_{i=1}^{k} N_i$.

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.

L'attaque de Wiener utilise la méthode des fractions continues pour obtenir d en ne connaissant uniquement la clé publique (N, e).

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

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

Puisque $ed \equiv 1 \pmod{\phi(N)}$, il existe k tel que $ed - k\phi(N) = 1$.
et donc :

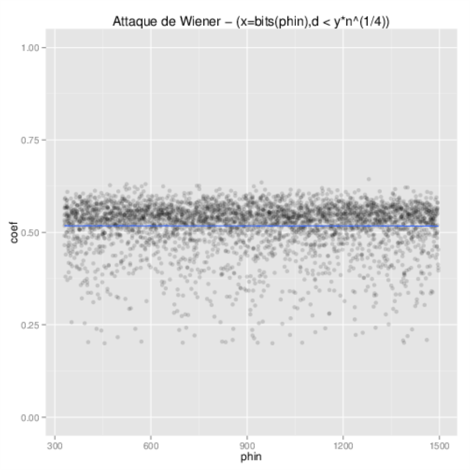
$$\left| \frac{e}{\phi(N)} - \frac{k}{d} \right| = \frac{1}{d\phi(N)}$$

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. Le développement en fractions continues de $\frac{e}{N}$ converge vers $\frac{k}{d}$ ce qui permet à partir des deux composantes de la clé publique d'obtenir la clé privée en temps linéaire.

BORNE DE WIENER

Th : Soit $N = pq$, avec $q < p < 2q$, et $d < \frac{1}{3}N^{\frac{1}{4}}$. Étant donnés (N, e) avec $ed \equiv 1(mod(\phi(N)))$, l'attaquant peut facilement retrouver d.

Nous nous sommes intéressé à cette borne et nous avons trouvé d pour $d < 0.52N^{\frac{1}{4}}$ en moyenne (échantillon de 2000 clés RSA).



Il existe plusieurs généralisations de l'attaque de Wiener, comme l'attaque de de Weger (2002).

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.

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.

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. 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 petites racines x_0 modulo b^m
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.

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.

Soient $b_1, b_2, \cdots, b_n$ une base d'un réseau L. 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 $\mathbb{R}^n$.

BASE RÉDUITE

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. $|b_1| \leq 2^{\frac{n-1}{4}} det(L)^{\frac{1}{n}}$

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 $\mathbb{Z}$

Posons $X = \frac{1}{2} N^{\frac{\beta^2}{\delta} - \epsilon}$.

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 $(1, x, x^2, \cdots, x^{m\delta+t-1})$.

On a une matrice triangulaire inférieure dont le déterminant est

$$det(L) = N^{\frac{1}{2} m(m+1)\delta} X^{\frac{1}{2} n(n-1)}$$

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 $|x_0| \leq X$.

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.

2 exemples

- Si on a une partie des bits de p on peut obtenir la factorisation de N.
- Si l'attaquant connaît le début du message, alors il peut décrypter le message en entier.

Nous avons étudié $OpenSSL$, un logiciel utilisant RSA utilisé par $\frac{2}{3}$ des sites-web.

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.

Il est possible de procéder à des attaques par factorisation pour trouver p et q depuis N.

Nous avons fait des tests avec l'algorithme $Msieve$ efficace N de taille inférieur à 110 chiffres ($\approx$ 350 bits).

ATTAQUE PAR FACTORISATION

Nous obtenons une clé 256bits en 2 minutes avec un ordinateur muni d'un processeur $i7$. Dès 350 bits, Msieve demande plus de 120heures pour le même ordinateur.

Aujourd'hui l'algorithme le plus efficace de factorisation est le 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.

Le record actuel de factorisation, le nombre $RSA-768$ (de taille 768bits) établi en 2009 par une équipe de 13 chercheurs a demandé plus de 2 ans de plusieurs ordinateurs en parallèle.
Le temps nécessaire correspond à 2000 années de calcul d'un simple $core$ d'un processeur AMD Opteron 2.2Ghz.

RSA-768=
1230186684530117755130494958384962720772853569595334792197
3224521517264005072636575187452021997864693899564749427740
6384592519255732630345373154826850791702612214291346167042
9214311602221240479274737794080665351419597459856902143413

Questions ?