Des éléments de solution sont affichés en vert.

Algorithme p-1 de Pollard

Nous avons vu en cours que la factorisation d'entiers RSA quelconques est un problème difficile, néanmoins solvable en temps *subexponentiel* à l'aide des algorithmes de crible. Toutefois, si l'entier RSA est mal généré, il existe des cas plus faciles.

Soit $N = PQ$ un module RSA où P et Q sont premiers et de même taille. Soit B une borne à choisir plus tard. L'algorithme $P-1$ de Pollard s'exécute selon les étapes suivantes :

1. Calculer $M = \prod_{\text{premiers } q \leq B} q^{\lfloor \log_q B \rfloor}$ (i.e., calculer le produit des puissances de nombres premiers inférieures à B)
2. Sélectionner a au hasard premier avec N
3. Calculer $G = GCD(a^M - 1, N)$
4. Si $G < N$ renvoyer G ; sinon renvoyer « échec »

Question 1. *On suppose que $P-1$ est B-ultrafriable (*B-powersmooth*), c'est-à-dire que toutes les puissances de nombres premiers p^ν dans sa décomposition en facteurs premiers sont plus petites que B. Montrer que M est multiple de $P-1$.*

Solution. *Question d'arithmétique triviale. Comme chaque produit p^ν divise M et qu'ils sont premiers entre eux, $P-1$ divise M.*

Question 2. *On suppose aussi que Q n'est* pas B-powersmooth*. Montrer que l'algorithme renvoie P avec bonne probabilité.*

Solution. *Par le théorème de Fermat. On sait que M s'écrit sous la forme $K(P-1)$ pour un entier K. On a $a^{K(P-1)} = 1 \mod P$ donc P divise G. Comme par hypothèse Q n'est pas B-powersmooth, un entier a pris au hasard ne vérifiera pas (avec bonne probabilité) la propriété du théorème de Fermat (c'est d'ailleurs le principe même du test de primalité de Fermat). Donc G ne sera pas multiple de Q. Par conséquent, $G = P$.*

Question 3. *Donner une borne sur la complexité de l'algorithme, à B fixé, puis à B variable.*

Solution. *Par le théorème des nombres premiers, il y a $\mathcal{O}(B/\log B)$ nombres premiers inférieurs à B. Le nombre M est donc le produit de $\mathcal{O}(B/\log B)$ nombres tous inférieurs à B. Il n'est pas nécessaire de le calculer directement, on peut calculer $a^M \mod N$ en faisant $\mathcal{O}(B/\log B)$ exponentiations modulaires successives.*

Chaque exponentiation modulaire est par un nombre $\leq B$, donc coûte $\mathcal{O}(\log B)$ multiplications modulo N. Chaque multiplication coûte de l'ordre de $\mathcal{O}(\log^2 n)$ opérations binaires en étant faite naïvement (et $\mathcal{O}(n \log n)$ asymptotiquement). Au total, pour calculer $a^M \mod N$ on a donc besoin de $\mathcal{O}(B \log^2 n)$ opérations (ou $\mathcal{O}(B)$ multiplications).

Le PGCD final est de coût négligeable.

Si B nous est inconnu, il faut le trouver en prenant une borne qui double à chaque fois et en exécutant l'algorithme jusqu'à ce qu'il fonctionne. Le coût sera donc :

$$\mathcal{O}\left(1 + 2 + \ldots + 2^{\lceil \log_2 B \rceil}\right) = \mathcal{O}(B)$$

multiplications.

Lors de la génération de « bons » nombres premiers on impose ainsi que $P-1$ ait au moins un « grand » facteur premier. Toutefois, la méthode de factorisation ECM (basée sur les courbes elliptiques) a rendu l'algorithme $P-1$ obsolète, et fonctionne aussi bien lorsque $P-1$ est powersmooth ou ne l'est pas.

Dans la méthode ECM, on utilise en effet le groupe des points d'une courbe elliptique quelconque définie sur $\mathbb{Z}_N$. Ce groupe est d'ordre variable, mais proche de N. Cette variabilité permet de tomber avec forte probabilité sur un ordre friable (smooth), contrairement à l'algorithme $P-1$ dans lequel le choix du groupe est contraint. On calcule donc les multiples d'un point de la courbe jusqu'à tomber sur un élément non-inversible, qui doit apparaître assez tôt à cause du petit théorème de Fermat. Cet algorithme est de complexité subexponentielle.

Autour de RSA

On rappelle le schéma de chiffrement RSA basique.

$\mathsf{KeyGen}(1^n)$ choisir un module N qui est le produit de deux premiers de n bits, avec deux entiers e et d tels que $ed = 1 \mod \phi(N)$. $\mathsf{pk} = (N, e)$; $\mathsf{sk} = (N, d)$
$\mathsf{Enc}(m \in \mathbb{Z}_N^*, (N, e))$ renvoie $c = m^e \pmod N$
$\mathsf{Dec}(c \in \mathbb{Z}_N^*, (N, d))$ renvoie $m = c^d \pmod N$

Nous avons déjà vu que ce RSA basique n'est pas IND-CPA. Dans cet exercice nous explorons quelques autres attaques sur ce schéma.

Question 4. *Soit $N = PQ$ un produit de deux premiers distincts. Montrer que si $\phi(N)$ et N sont connus, alors on peut retrouver p, q en temps polynomial.*

Solution. *Trivial : $\phi(N) = (P-1)(Q-1)$.*

Question 5. *Montrer que si $m \in [0, N^{1/e}]$ alors on peut facilement décrypter (= retrouver le message sans connaître la clé privée, par opposition à* déchiffrer*).*

Solution. *L'équation $c = m^e \pmod N$ tient alors dans les entiers. On peut calculer la racine !*

Question 6. *Une* racine de l'unité *modulo N est un entier x tel que $x^2 = 1 \mod N$.*

1. *Combien y a-t-il de racines de l'unité modulo N ?*
2. *Supposons que l'on connaisse (N, e, d) (mais pas la factorisation de N). Montrer qu'on peut calculer une racine de l'unité modulo N. On admet qu'elle est non-triviale avec bonne probabilité.*
3. *En déduire qu'on peut factoriser N.*

Solution. *1. Il y a quatre racines de l'unité modulo N. Cela vient du CRT : si $x^2 = 1 \mod N$ alors $x^2 = 1 \mod P$ et $x^2 = 1 \mod Q$. Donc soit $x = 1 \mod P$ et $x = 1 \mod Q$ (ou -1) et dans ce cas $x = \pm 1$ (racines triviales). Soit $x = 1 \mod P$ et $x = -1 \mod Q$ (ou l'inverse) et dans ce cas x est une racine non-triviale.*

2. Calculer $k = de - 1$, on sait que k est un multiple de $\phi(N)$ par définition. De plus, $\phi(N) = (P-1)(Q-1)$ est pair. Donc $k = 2^t r$ où r est impair.

On sait que pour tout g, $g^k = g^{2^t r} = 1 \mod N$, donc $g^{2^{t-1} r}$ est une racine de l'unité modulo N. Comme g a été pris au hasard, on s'attend à tomber sur une racine non-triviale avec bonne probabilité.

3. On est donc capable de calculer $\pm x$ (en prenant g au hasard, on s'attend à tomber sur x avec bonne probabilité). On peut ensuite calculer $GCD(x-1, N)$ qui donne un facteur premier de N. Tout ce calcul s'effectue en temps polynomial.

Question 7. *Fixons un module N et supposons qu'un serveur centralisé donne aux utilisateurs des paires (e_1, d_1) et (e_2, d_2) formant des clés RSA valides (exposants privés et publics). Pourquoi est-ce une mauvaise idée ?*

Solution. *Comme Alice possède (e_1, d_1) elle peut factoriser N, et donc retrouver les exposants secrets des autres clés.*

Question 8. *Soit $(N_1, e), \ldots, (N_e, e)$ les clé publiques de e utilisateurs différents. Un même message m est chiffré e fois, avec chacune de ces clés publiques. Montrer qu'un attaquant peut retrouver m à partir de l'observation des chiffrés $c_i := \mathsf{Enc}(m, (N_i, e))$.*

Solution. *Comme on a $m^e \pmod{N_i}$ pour tout i, à l'aide du CRT on calcule $m^e \pmod{N_1 \cdots N_e}$. (Notez que pour que cette attaque soit efficace, on a besoin que e soit polynomial en n).*

Comme $m < N_i$ pour tout i, on note que $m^e < N_1 \cdots N_e$. Par conséquent on a obtenu la valeur de m^e en tant qu'entier. On peut donc en déduire m.

Question 9. *On essaie maintenant d'éviter l'attaque de la question précédente. On a $m < \sqrt{N_i}$ mais on force chaque utilisateur à utiliser une modification de son message m, sous la forme d'un décalage δ_i connu. L'attaquant n'observe donc plus que les chiffrés de $m + \delta_1$, $m + \delta_2$, …, $m + \delta_e$.*

On admet le théorème de Coppersmith :

Theorem 1. *Soit $f \in \mathbb{Z}[X]$ un polynôme unitaire de degré e et N un entier. S'il existe une racine x_0 de f modulo N telle que $|x_0| \leq N^{1/e-\varepsilon}$, alors il est possible de retrouver x_0 en temps polynomial en $\log N$ et $1/\varepsilon$.*

Montrer comment retrouver m.

Solution. *Soient $f_1, \ldots, f_e$ les fonctions : $f_i(m) = (m + \delta_i)^e - c_i$ (qui sont donc des polynômes de degré e en m).*

On a pour tout i : $f_i(m) = 0 \pmod{N_i}$. Soit $N = N_1 \cdots N_e$. Supposons que les N_i sont tous premiers entre eux (sinon on a déjà gagné !).

D'après le CRT, comme les N_i sont tous premiers entre eux, il existe des nombres t_i tels que : $t_i = 1 \pmod{N_i}$ et $t_i = 0 \pmod{N_j}$ pour tout $j \neq i$. En d'autres termes t_i est l'image de $(0, \ldots, 1, \ldots 0)$ par l'isomorphisme entre $\prod_i \mathbb{Z}_{N_i}$ et $\mathbb{Z}_N$.

On définit la fonction :

$$g(x) := \sum_i t_i f_i(x)$$

et on remarque que $g(m) = 0 \pmod{N}$ par définition des t_i. De plus g est un polynôme en x de degré e, et le coefficient de x^e est $t_1 + \ldots + t_e$. On peut donc multiplier par l'inverse de $t_1 + \ldots + t_e$ modulo N (en supposant qu'il existe) pour obtenir un polynôme unitaire de degré e.

Comme on a la garantie que m est relativement petit, on peut utiliser le théorème de Coppersmith pour conclure.

Fonction Indicatrice d'Euler

On rappelle que l'indicatrice d'Euler est définie par $\phi(N) = |\mathbb{Z}_N^*|$, l'ordre du groupe $\mathbb{Z}_N^*$. Dit autrement, c'est le nombre d'entiers de $[1; N]$ qui sont premiers avec N.

Question 10. *Soit p un nombre premier, montrer que $\phi(p) = p - 1$.*

Question 11. *Soient p, q premiers entre eux. Montrer que $\phi(qp) = \phi(p)\phi(q)$.*

Question 12. *Soit p un premier et $e \geq 1$ un entier. Montrer que $\phi(p^e) = p^{e-1}(p-1)$.*

Question 13. *Soit $N = \prod_i p_i^{c_i}$ où les p_i sont des premiers distincts, $c_i \geq 1$. Montrer que $\phi(N) = \prod_i p_i^{c_i - 1}(p_i - 1)$;*