MATHÉMATIQUES DISCRÈTES (5)

Michel Rigo

http://www.discmath.ulg.ac.be/

Année 2007–2008

Fonction à sens unique

fonction à sens unique telle que $f(x)$ est “facile” à calculer pour tout $x \in A$ et $f^{-1}(y)$ est “difficile à calculer”.

La notion de “facilité” fait référence aux ressources à mettre en oeuvre pour effectuer les calculs de f et f^{-1} (complexité temporelle).

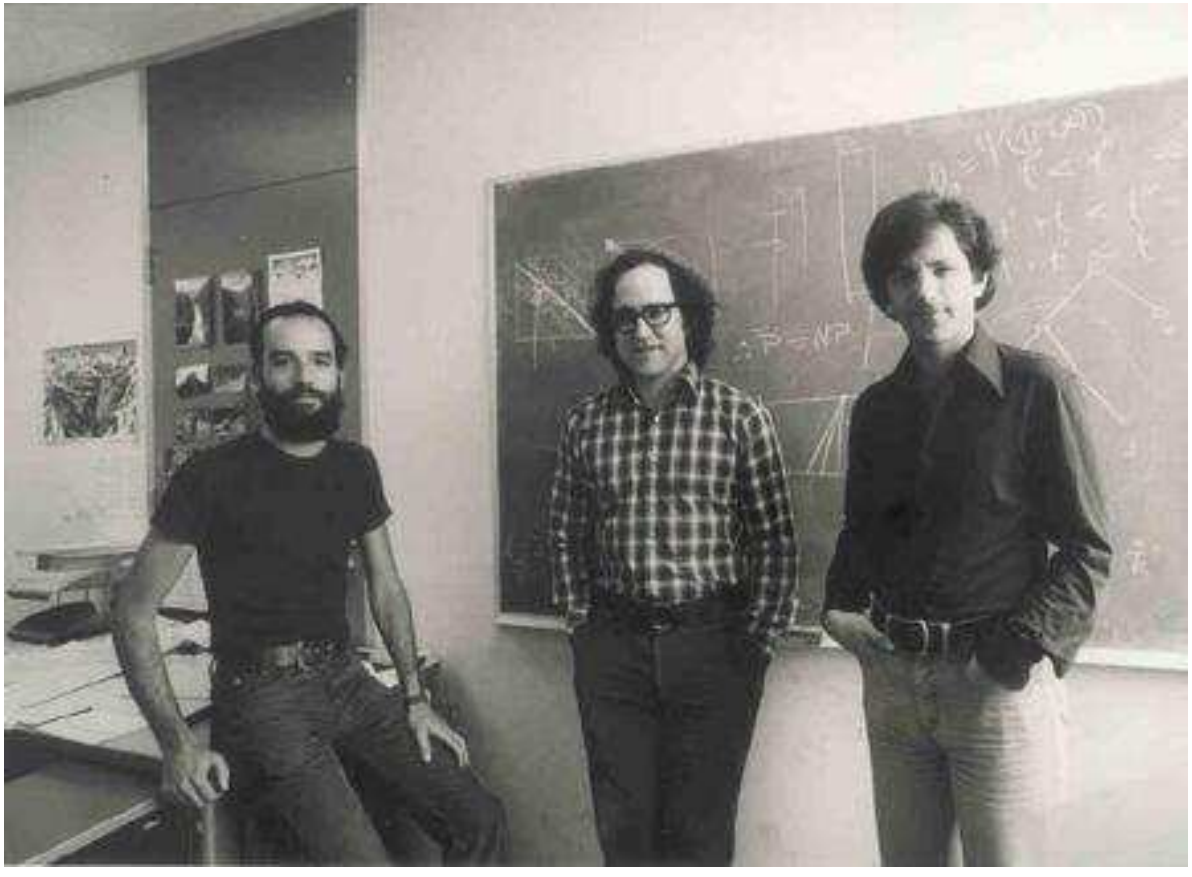
Fonction à sens unique

Dumbeldore, A.	03189228
⋮	⋮
Fudge, C.	02162276
Ganger, H.	04122782
⋮	⋮
Hagrid, R.	02765100
Potter, H.	04378128
⋮	⋮

Remarque

Fonction à sens unique “pure” / Fonction à trappe cachée

RSA



Ron **R**ivest, Adi **S**hamir et Leonard **A**dleman (1977)

RSA

Bob choisit deux grands nombres premiers distincts p et q.

$$n = p.q.$$

Puisque p et q sont premiers,

$$\varphi(n) = (p-1).(q-1).$$

Bob choisit à présent deux nombres e et d tels que

$$d.e \equiv 1 \mod \varphi(n).$$

Pour ce faire, il choisit e tel que

$$1 < e < \varphi(n) \quad \text{et} \quad \text{pgcd}(e, \varphi(n)) = 1.$$

Ensuite, Bob obtient d grâce à l'algorithme d'Euclide étendu.

Bob publie n, e et conserve secret les autres éléments $d, p, q, \varphi(n)$.

RSA

e (resp. d) l'*exposant de chiffrement* (resp. *de déchiffrement*).
$k = (e, n)$ est la clé du système.

Si l'ensemble des textes clairs est $\mathcal{P} = \mathbb{Z}_n$, alors

$$\forall x \in \mathbb{Z}_n,\ e_k(x) := x^e \mod n.$$

PROPOSITION

Si $k = (e, n)$ et si $y = e_k(x)$, alors

$$y^d \mod n = x.$$

RSA

Petit théorème de Fermat : si $\mathrm{pgcd}(x, n) = 1$, alors

$$x^{\varphi(n)} \equiv 1 \mod n.$$

Ici, $y = x^e$. Ainsi, il existe $\alpha \in \mathbb{N}$ tel que

$$y^d = (x^e)^d = x^{ed} = x^{1+\alpha\,\varphi(n)}$$

car $ed \equiv 1 \mod \varphi(n)$. Si x est premier avec n, le résultat est OK en appliquant le petit théorème de Fermat.

Supposons x non premier avec n. Il est cependant premier avec p ou q car sinon, il serait multiple de n, or $x < n$.

Supposons dès lors, x premier avec p mais pas avec q. Par le petit théorème de Fermat,

$$x^{ed} = x\,(x^{p-1})^{\alpha(q-1)} \equiv x \mod p$$

car $x^{p-1} \equiv 1 \mod p$. On a aussi trivialement

$$x^{ed} \equiv x \mod q$$

car les deux membres sont congrus à zéro modulo q.

De ces deux égalités, puisque p et q sont premiers et distincts, on en tire que

$$x^{ed} \equiv x \mod n.$$

En effet, si $y \equiv x \mod p$ et $y \equiv x \mod q$,
$y = x + m_1 p$ et $y = x + m_2 q$.
De là, $m_1 p = m_2 q$.
Puisque p et q sont premiers et distincts, $p | m_2$ et $q | m_1$.
Il existe α_1 et α_2 tels que $m_1 = \alpha_1 q$ et $m_2 = \alpha_2 p$.
De $m_1 p = m_2 q$, on tire $\alpha_1 pq = \alpha_2 pq$. D'où $\alpha_1 = \alpha_2$ et
$y = x + \alpha_1 pq = x + \alpha_1 n$.

RSA

En résumé, on a les données suivantes :

- Bob publie : $k = (e, n)$,
- Bob conserve secret : d, p, q, $\varphi(n)$,
- fonction de chiffrement : $e_k(x) = x^e \mod n$,
- fonction de déchiffrement : $d_k(y) = y^d \mod n$.

Remarque

Supposons qu'Alice et Bob désirent converser par courrier électronique en utilisant le RSA. Dans ce cas, Alice construit des nombres n_A, e_A, d_A et publie $k_A = (e_A, n_A)$. Bob fait de même. Il construit n_B, e_B, d_B et publie $k_B = (e_B, n_B)$. Alice (resp. Bob) utilisera alors e_{k_B} (resp. e_{k_A}) pour chiffrer des messages destinés à Bob (resp. Alice). En pratique, on peut donc supposer disposer d'une sorte d'annuaire reprenant les utilisateurs et leur clé publique respective.

Exemple

$$p = 11,\ q = 23,\ n = p.q = 253,\ e = 3.$$

Puisque $220 = 3.73 + 1$, on en déduit que l'exposant de déchiffrement d est tel que

$$3^{-1} = -73 \mod 220 = 147 \mod 220.$$

Bob publie $k = (3, 253)$. Si Alice veut envoyer le texte clair $x = 165$ à Bob, elle calcule

$$e_k(x) = 165^3 \mod 253 = 110.$$

Si Bob reçoit le message $y = 110$, il lui suffit de calculer

$$d_k(y) = 110^{147} = 110^{128}.110^{16}.110^{2}.110^{1} \mod 253 = 165.$$

$p, q \sim 2^{512}$.

$$\lfloor \log_{10} 2^{512} \rfloor = \left\lfloor 512 \underbrace{\log_{10} 2}_{\sim 0,3} \right\rfloor = 154$$

Remarque

$N = 26, 32$ ou 256 est bien trop petit par rapport à $n \sim 2^{1024}$. Si on utilisait comme ensemble de textes clairs $\mathbb{Z}_N$ au lieu de $\mathbb{Z}_n$, alors une recherche exhaustive de toutes les images $x^e \mod n$ pour $x \in \mathbb{Z}_N$ réduit à néant la sécurité du RSA.

unité de base, les blocs de k symboles consécutifs de l'alphabet Σ avec k défini par

$$k := \left\lfloor \log_N n \right\rfloor,$$

autrement dit, $N^k \leq n < N^{k+1}$.

k éléments consécutifs de $\mathbb{Z}_N$ correspondent à un nombre $x \in \mathbb{Z}_{N^k}$ écrit en base N et compris entre 0 et $N^k - 1$.

k est la plus grande valeur possible permettant ce codage.
Si $x \in \mathbb{Z}_{N^k}$, le chiffrement de x est donné par

$$y = x^e \mod n$$

qui est un entier $y < n$ et $n < N^{k+1}$.

La représentation de y en base N peut être de longueur $k + 1$.
Les blocs de k éléments consécutifs de $\mathbb{Z}_N$ sont envoyés de manière injective sur des blocs de longueur $k + 1$.

Exemple

$n = 253$ et $e = 3$. L'alphabet $\Sigma = \{a, b, c, d\}$ et le codage $a \to 0$, $b \to 1$, $c \to 2$, $d \to 3$). $N = \#\Sigma = 4$ donc

$$k = \left\lfloor \log_4 253 \right\rfloor = 3$$

car $4^3 = 64$ et $4^4 = 256$. Alice désire envoyer à Bob le message

$$bccadb$$

Tout d'abord, le premier bloc de longueur 3, *bcc*, correspond à

$$1.4^2 + 2.4 + 2.1 = 26 < 64$$

Exemple

et son chiffrement est donné par

$$26^3 \mod 253 = 119 = 1.4^3 + 3.4^2 + 1.4^2 + 3.4^0$$

qui correspond au mot *bdbd*. Le second bloc de texte clair, *adb*, correspond à

$$0.4^2 + 3.4 + 1 = 13$$

et son chiffrement 13^3
$\mod 253 = 173 = 2.4^3 + 2.4^2 + 3.4^1 + 1.4^0$ fournit le mot *ccdb*.

Alice transmet le texte chiffré *bdbdccdb* Quant à Bob, il sait qu'il doit redécouper le texte chiffré en blocs de longueur $k + 1 = 4$ avant déchiffrement.

Question

Comment Bob peut-il être certain que le message provient bien d'Alice et non pas d'Oscar se faisant passer pour Alice ?

n_A, e_A, d_A n_B, e_B, d_B

Idée :

$$Alice : x \longrightarrow x^{d_A} \longrightarrow (x^{d_A})^{e_B}$$

$$Bob : (x^{d_A})^{e}_{B} \longrightarrow ((x^{d_A})^{e_B})^{d_B} = x^{d_A} \longrightarrow (x^{d_A})^{e_A} = x$$

En pratique

Si $n_A > n_B$, alors des $x^{d_A} \mod n_A$ distincts peuvent être égaux modulo n_B et le protocole proposé ne serait dès lors plus injectif.

Alice et Bob décident d'une valeur commune t.

Ils construisent chacun deux cryptosystèmes RSA

- Alice publie $(e_{A,s}, n_{A,s})$ et $(e_{A,c}, n_{A,c})$
- Bob publie $(e_{B,s}, n_{B,s})$ et $(e_{B,c}, n_{B,c})$

La construction suppose que

$$n_{A,s} < t < n_{A,c} \quad \text{et} \quad n_{B,s} < t < n_{B,c}.$$

De cette manière, Alice signe son message en utilisant l'exposant $d_{A,s}$ pour obtenir un nombre inférieur à $n_{A,s}$. Enfin, le message est chiffré avec les clé $(e_{B,c}, n_{B,c})$ et on a bien $n_{A,s} < n_{B,c}$.

REMARQUE

Pas raisonnable qu'Alice et Bob emploient le même n, car alors, connaissant tous deux la factorisation de n, ils pourraient déchiffrer les messages destinés à l'autre utilisateur.

Problème de signature cachée

Pour tous $x_1, x_2 \in \mathbb{Z}_n$,

$$(x_1 x_2)^d = x_1^d \, x_2^d \mod n \quad (d = d_A).$$

Si Alice signe les messages $x = x_1 x_2$ et x_2, elle signe également, sans pour autant le vouloir, le message x_1.

En effet, si on dispose de x^d et de x_2^d, on trouve

$$x_1^d = x^d (x_2^d)^{-1} \mod n$$

et ce même sans avoir connaissance de l'exposant d.

A condition que x_2^d soit inversible (très probable : $\varphi(n)/n = (p-1)(q-1)/pq$).

Construire des fonctions de hachage t.q.

$$\mu(x_1 x_2) \neq \mu(x_1)\mu(x_2)$$

et qu'il est "difficile" de trouver $x_1 \neq x_2$ t.q. $\mu(x_1) = \mu(x_2)$.

On applique la fonction de hachage μ au texte T à signer dans son intégralité et le résultat est un texte $\mu(T)$ de taille fixe (souvent 160 bits).

Uniquement le résultat est signé : $(\mu(T))^d \mod n$.

La fonction de hachage μ n'est pas injective, mais on cherche à minimiser la probabilité que deux textes distincts ayant du sens possèdent la même image par μ.

Permet aussi de diminuer le nombre de calculs à effectuer pour une signature.

But

Trouver d (resp. $\varphi(n)$) est aussi difficile que factoriser n.

Plus précisément, Si on dispose de d (resp. de $\varphi(n)$) , alors on peut construire un algorithme permettant de factoriser n.

Si d est connu

On pose

$$s = \max\{t \in \mathbb{N} \mid 2^t \text{ divise } ed - 1\} \quad \text{et} \quad k = \frac{ed-1}{2^s}.$$

càd, 2^s est la plus grande puissance de 2 qui divise $ed - 1$.

Lemme

Si a est premier avec n, l'ordre de a^k dans le groupe multiplicatif $\mathbb{Z}_n^*$ est de la forme 2^i pour un $i \in \{0, \ldots, s\}$.

Puisque a est premier avec n, le petit théorème de Fermat stipule que

$$a^{ed-1} \equiv 1 \mod n.$$

Par définition de k, on a $ed - 1 = k2^s$ et donc

$$(a^k)^{2^s} \equiv 1 \mod n.$$

Donc l'ordre de a^k divise 2^s.

Proposition

Soit a premier avec n. Si les ordres de a^k modulo p et modulo q diffèrent, alors il existe $t \in \{0, \ldots, s-1\}$ tel que

$$1 < \text{pgcd}(a^{2^t k} - 1, n) < n.$$

Par le lemme précédent, il existe $i \leq s$ tel que

$$(a^k)^{2^i} = 1 + \alpha n = 1 + (\alpha p)q.$$

2^i est un multiple de l'ordre de a^k modulo q
$\rightarrow$ l'ordre de a^k modulo q est de la forme 2^t
$\rightarrow$ l'ordre de a^k modulo p est de la forme $2^{t'}$.

Par hypothèse, les ordres diffèrent, supp. $t < t' \leq s$. De là, on a

$$(a^k)^{2^t} \equiv 1 \mod q \quad \text{et} \quad (a^k)^{2^t} \not\equiv 1 \mod p.$$

Ainsi, $(a^k)^{2^t} - 1$ est un multiple de q mais pas de p et

$$\text{pgcd}(a^{2^t k} - 1, n) = q.$$

Algorithme (factoriser n)

Choisir aléatoirement $a \in \{1, \ldots, n-1\}$.
Calculer $g = \mathrm{pgcd}(a, n)$.
  Si $g > 1$, on a trouvé un facteur de n.
  Si $g = 1$,
    pour $t = s-1, s-2, \ldots$
      calculer $g' = \mathrm{pgcd}(a^{2^t k} - 1 \mod n, n)$
    jusqu'à obtenir $g' > 1$ ou arriver à $t = 0$.
  Si $g' > 1$, on a trouvé un facteur de n.
  Sinon, $t = 0$, recommencer avec un nouveau choix de a.

Question

Déterminer parmi les $\varphi(n)$ candidats "a" premiers avec n combien sont tels que a^k possède des ordres différents modulo p et q.

Résultat admis

Le nombre d'entiers $a < n$, premiers avec n et tels que a^k possède des ordres différents modulo p et q est $\geq \varphi(n)/2$.

Corollaire

La probabilité de tirer consécutivement k nombres "a" au hasard dans l'algorithme et qu'aucun de ceux-ci ne permette de factoriser n est inférieure à

$$2^{-k}.$$

Si $k = 10$, $1 - 2^{-k} = 1023/1024 \sim 0,999$.

Si $\varphi(n)$ est connu

$$n = p.q \quad \text{et} \quad \varphi(n) = (p-1)(q-1).$$

En remplaçant q par n/p dans la seconde équation, on obtient

$$\varphi(n) = (p-1)(\frac{n}{p} - 1)$$

c'est-à-dire,

$$p^2 + p\,[\varphi(n) - n - 1] + n = 0.$$

Puisque $\varphi(n)$ est connu, il s'agit d'une simple équation du second degré permettant de retrouver p.

RAPIDITÉ DU RSA

Si $e \simeq n = p.q$ et si l'on suppose que son écriture binaire contient autant de bits 0 que de 1, alors travaillant avec un entier $n \sim 2^{1024}$, pour réaliser une exponentiation modulaire, on a besoin de 1024 élévations au carré et 512 multiplications modulo n.

On peut estimer le RSA de 1000 à 10000 fois plus lent que le DES.

RSA pour échanger en toute sécurité une clé d'un cryptosystème à clé secrète.

QUELQUES ÉLÉMENTS DE SÉCURITÉ

1. p et q ne doivent pas être “petits”, ni provenir d’une table.

Recherche exhaustive ou tester les nombres de la table

EXEMPLE

Sous Mathematica , tester si 15485863 est divisible par l’un des cent mille premiers nombres premiers prend sur un pentium III à 600 Mhz un peu moins de $3,4$ secondes.

2. p et q ne doivent pas être trop proches.

si $p > q$, alors $(p-q)/2$ est petit et $(p+q)/2$ est un peu plus grand que $\sqrt{n}$. Par ailleurs, il est clair que

$$\left(\frac{p+q}{2}\right)^2 - n = \left(\frac{p-q}{2}\right)^2.$$

Ainsi, il suffit de considérer successivement tous les entiers $x > \sqrt{n}$ et pour ceux-ci, de calculer $x^2 - n$ jusqu'à obtenir un carré parfait y^2. Dans ce cas, $x^2 - y^2 = n$ et on trouve $p = x + y$ et $q = x - y$.

EXEMPLE, $n = 97343$

On a $\sqrt{n} > 311$ et $312^2 - n = 1$. Par conséquent, $n = 311.313$.

Exemple, $n = 239812789091371$

On a $\sqrt{n} > 15485889$ et

$$15485890^2 - 239812789091371 = 729 = 27^2.$$

Donc,

$$239812789091371 = (15485890 + 27)(15485890 - 27).$$

Il s'agit en fait du produit des millionième et $(10^6 + 2)$-ième nombres premiers.

3. $p-1$ et $q-1$ ne doivent pas avoir un large facteur commun (i.e., un pgcd trop grand).

Sinon $u = \mathrm{ppcm}(p-1)(q-1)$ est “relativement” petit en comparaison de $\varphi(n)$.

Tout inverse de e modulo u peut être utilisé comme exposant de déchiffrement.

En effet, si $z.e \equiv 1 \mod u$, alors
$z.e = 1 + \alpha(p-1) = 1 + \beta(q-1)$ et il suffit d’adapter la seconde partie de la preuve... (déchiffrement RSA)

Bien que u soit inconnu, il peut être suffisamment petit pour procéder à une recherche exhaustive. Ainsi, il suffit de tester des candidats u successifs et pour chacun d’entre eux calculer l’inverse de e modulo u.

si $z.e \equiv 1 \mod u$, $z.e = 1 + \alpha(p-1) = 1 + \beta(q-1)$.

Soit $x \in \mathbb{Z}_n$. On a

$$(x^e)^z = x^{ez} = x(x^\alpha)^{p-1}.$$

Si x est premier avec p, alors x^α aussi et

$$x.(x^\alpha)^{p-1} \equiv x.1 \mod p.$$

Si x est un multiple de p, les deux membres ci-dessus sont congrus à 0 modulo p.
Pour tout $x \in \mathbb{Z}_n$, $(x^e)^z \equiv x$ modulo p.

raisonnement idem pour q. On en tire que pour tout $x \in \mathbb{Z}_n$, $(x^e)^z \equiv x$ modulo n, z est bien un exposant de déchiffrement.

Exemple

Soient $p = 61$, $q = 181$, $n = 11041$ et $e = 4013$. Ici, $\varphi(n) = 60.180 = 10800$. Ici, $\text{ppcm}(p-1, q-1) = q - 1$.

Le ppcm étant pair, il suffit de tester successivement pour candidats $u = 2, 4, \ldots, 180$.

Après 90 tests, Oscar est en mesure de construire un exposant de déchiffrement.

Exemple

Soient $p = 61$, $q = 179$, $n = 10919$ et $e = 4013$. Ici, $\varphi(n) = 60.178 = 10680$. Du point de vue de l'ordre de grandeur, les différences sont minimes. Néanmoins, ici

$$\text{ppcm}(60, 178) = 60.89 = 5340$$

la recherche exhaustive nécéssite 2770 essais.

4. Si $\varphi(n)$ ne possède que de petits facteurs premiers, i.e.,

$$\varphi(n) = p_1^{\alpha_1} \cdots p_k^{\alpha_k}, \quad p_i \leq r$$

où r est une borne suffisamment petite ($p_1, \ldots, p_k$ représentent tous les nombres premiers $\leq r$ et $\alpha_i = 0$ si p_i n'apparaît pas).

$$p_i^{\lfloor \log_{p_i} n \rfloor}$$

est la plus grande puissance de p_i qui peut éventuellement diviser $\varphi(n)$ (en effet, $\varphi(n) < n$ et ne connaissant pas la valeur de $\varphi(n)$, on utilise alors n comme estimation).

On décide de passer en revue les candidats potentiels pour $\varphi(n)$: ils sont de la forme

$$u = p_1^{\beta_1} \cdots p_k^{\beta_k} \text{ avec } \beta_i \leq \lfloor \log_{p_i} n \rfloor, \forall i \leq k.$$

Pour chaque candidat potentiel u :
si $(u+1)/e$ est un entier d', on essaye d' comme exposant de déchiffrement (c'est assez naturel, car alors $ed' = 1 + u$).

Si r n'est pas trop grand, les candidats potentiels à tester ne sont pas trop nombreux et une recherche exhaustive est réalisable.

Exemple

Soit n le produit des 100- et 101-ièmes nombres premiers,

$$n = 295927 \text{ et } \varphi(n) = 2^3.3^4.5.7.11^0.13$$

et $\log_2 n = 18$, $\log_3 n = 11$, $\log_5 n = 7$, $\log_7 n = 6$, $\log_{11} n = 5$, $\log_{13} n = 4$.

Les candidats u sont (en considérant $r = 13$) de la forme

$$2^{i_1}.3^{i_2}.5^{i_3}.7^{i_4}.11^{i_5}.13^{i_6}$$

avec $0 \leq i_1 \leq 18$, $0 \leq i_2 \leq 11$, $0 \leq i_3 \leq 7$, $0 \leq i_4 \leq 6$, $0 \leq i_5 \leq 5$ et $0 \leq i_6 \leq 4$. Le nombre total de candidats est donc

$$19.12.8.7.6.5 = 383040.$$

Exemple

Par contre, pour n, produit des 101- et 102-ièmes nombres premiers, on a

$$n = 304679 \text{ et } \varphi(n) = 2^3.3.7.13.139.$$

Ici, le plus grand facteur premier apparaissant dans la décomposition de $\varphi(n)$ est 139 (qui est le 34-ième nombre premier) et si on procède comme ci-dessus, les candidats u sont de la forme

$$2^{i_1}.3^{i_2}.5^{i_3}.7^{i_4} \ldots 131^{i_{32}}.137^{i_{33}}.139^{i_{34}}$$

et le nombre total de candidats est

$$5149048008867840.$$

5. Si les exposants e et d sont petits, il existe également des attaques possibles du RSA.

Compromis entre sécurité et temps de calcul (par exemple, pour un système RSA placé sur une carte à puce pour laquelle les ressources de calcul sont limitées)

Génération de grandes nombres premiers

Trouver un nombre premier de k bits écrit en base 2, on considère un vecteur appartenant à $\{0,1\}^k$: $1 * \cdots * 1$.

Tests de primalité

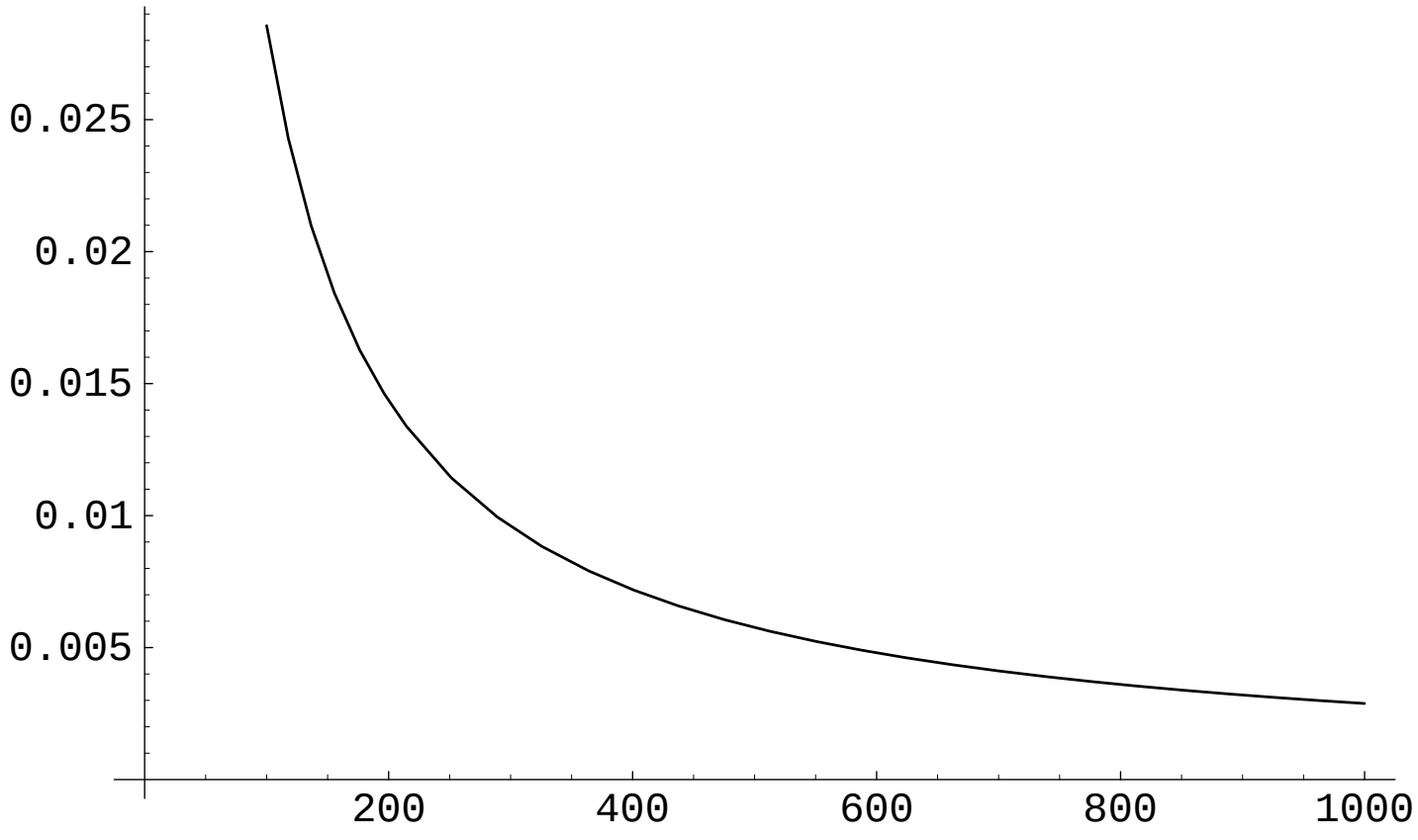
Approximation du nombre de tirages à réaliser pour obtenir un nombre premier de k bits

le nombre de nombres premiers de k bits est

$$\pi(2^k) - \pi(2^{k-1}) \sim \frac{2^k}{\ln 2^k} - \frac{2^{k-1}}{\ln 2^{k-1}}$$

et le nombre total d'entiers impairs de k bits est 2^{k-2}.

$k = 100$, proba proche de $0,0285$ et pour $k = 512$, $0,0056$.


0.025
0.02
0.015
0.01
0.005
200
400
600
800
1000

Le petit théorème de Fermat fournit un test de primalité probabiliste.

Si m est premier, alors $x^{m-1} \equiv 1 \mod m$. Donc, si pour x premier avec m, on a $x^{m-1} \not\equiv 1 \mod m$, alors on en conclut directement que m n'est pas premier.

Test de primalité de Fermat, donnée m

Choisir aléatoirement x tel que $1 \leq x < m$.
Calculer $g = \mathrm{pgcd}(x, m)$.
- Si $g > 1$, alors m n'est pas premier.
- Sinon, calculer $t = x^{m-1} \mod m$.
  - Si $t \not\equiv 1$, alors m est composé.
  - Si $t \equiv 1$, alors m est peut-être premier.

La réponse est : soit “*m* est composé” (on ne dispose pas de la factorisation de *m* mais on est certain que ce nombre n’est pas premier)
soit “*m* est peut-être premier”. En effet, pour *m* composé, il existe des entiers *x* tels que $x^{m-1} \equiv 1 \mod m$. La détermination du caractère composé de *m* dépend de l’entier *x* tiré aléatoirement.

Exemple

Si x est tq. $1 \leq x < m$, $\text{pgcd}(x, m) = 1$ et $x^{m-1} \equiv 1 \mod m$, alors x est témoin de la primalité de m ou
m est un nombre pseudo-premier relativement à la base x.

"Au plus on a de témoins, au plus on y croît !"

Exemple

Le nombre 91 est composé, $91 = 13.7$.

$$2^{90} = 2^{64}.2^{16}.2^{8}.2^{2} \equiv 64 \mod 91.$$

Ainsi, si on tire aléatoirement $x = 2$, on montre que 91 n'est pas premier : "2 n'est pas témoin de la primalité de 91".

Par contre, $3^{90} = 3^{64}.3^{16}.3^{8}.3^{2} \equiv 1 \mod 91$.

Dès lors, si on tire $x = 3$, on n'est pas en mesure de prouver la non primalité de 91 : "91 est pseudo-premier relativement à 3".

Proposition

Considérons un entier m. Soit tous, soit au plus la moitié des entiers x tels que $1 \leq x < m$ et $\mathrm{pgcd}(x, m) = 1$ sont témoins de la primalité de m.

Remarque

Dans la situation où un entier composé m est tel qu'au plus la moitié des entiers x sont témoins de la primalité de m, si on effectue k tests consécutifs, la probabilité que m soit considéré comme pseudo-premier à l'issue de ces k tests est $\leq 2^{-k}$.

Supposons que x n'est pas témoin de la primalité de m, i.e.,

$$x^{m-1} \not\equiv 1 \mod m.$$

Soient $w_1, \ldots, w_t$ tous les témoins (distincts) de primalité de m. Posons

$$u_i = x\, w_i \mod m.$$

Les u_i sont tous distincts car x est premier avec m donc inversible modulo m. Si $u_i = u_j$ avec $i \neq j$, alors on en déduirait que $w_i = w_j$, ce qui est impossible. De plus, $1 \leq u_i < m$ et $\mathrm{pgcd}(u_i, m) = 1$ car x et les w_i, par définition, sont premiers avec m. Les u_i ne sont pas témoins de la primalité de m. En effet,

$$u_i^{m-1} = \underbrace{x^{m-1}}_{\not\equiv 1} \underbrace{w_i^{m-1}}_{\equiv 1} \not\equiv 1 \mod m.$$

Conclusion : s'il existe x non témoin de la primalité de m, alors il y a au moins autant de nombres u_i non témoins que de w_i qui sont témoins.

Définition

Il existe des nombres composés m pour lesquels tout $x < m$ et premier avec m est témoin de primalité de m, i.e., $x^{m-1} \equiv 1 \mod m$. Un tel nombre est appelé nombre de Carmichael.

Test de Fermat :-(

Le seul espoir bien faible de détecter la non primalité d'un tel nombre m est de tirer au hasard un nombre x ayant un facteur commun avec m.

Proposition

Un nombre composé m est de Carmichael SSI il n'est divisible par aucun carré (square-free ou quadratfrei) et pour tout p premier divisant m, $p-1$ divise $m-1$.

Exemple

Le nombre $561 = 3.11.17$ est un nombre de Carmichael (c'est même le plus petit). En effet, $560 = 280.2$, $560 = 56.10$ et $560 = 35.16$.

On pourrait vérifier que tout $x < m$ et premier avec m est tel que $x^{m-1} \equiv 1 \mod m$.

$\Rightarrow$ Soit $m \geq 3$ un nombre de Carmichael.
Soit p un diviseur premier de m.
Soit a racine primitive modulo p (i.e., un générateur de $\mathbb{Z}_p^*$) qui est premier avec m. Un tel a existe.

$\mathbb{Z}_p$ est un champ, $\mathbb{Z}_p^*$ possède $\varphi(p-1)$ générateurs. Soit u un de ces générateurs modulo p.
Si $m = p^\alpha q_1^{\beta_1} \cdots q_r^{\beta_r}$, alors a s'obtient par exemple comme solution du système

$$\begin{cases} a \equiv u & \mod p, \\ a \equiv 1 & \mod q_1, \\ \vdots & \\ a \equiv 1 & \mod q_r. \end{cases}$$

Si a est solution de ce système, il est nécessairement premier avec m puisqu'il n'a aucun facteur commun avec m. Le théorème des restes chinois assure l'existence d'un tel a.

Théorème des restes chinois

Si $m_1, \ldots, m_n$ sont deux à deux premiers, le système

$$x \equiv a_1 \mod m_1, \ldots, x \equiv a_n \mod m_n$$

possède une unique solution modulo $m = m_1 \cdots m_n$. Si, pour tout $i = 1, \ldots, n$, on pose $M_i = m/m_i$ et $y_i = M_i^{-1} \mod m_i$, alors cette solution est donnée par

$$x = \left(\sum_{i=1}^{n} a_i y_i M_i \right) \mod m.$$

Puisque a est premier avec m (nombre de Carmichael), on a $a^{m-1} \equiv 1 \mod m$ et donc $a^{m-1} \equiv 1 \mod p$ (car $p|m$). a racine primitive mod p, son ordre modulo p est $p-1$ donc $p-1$ divise $m-1$.

p^2 ne divise pas m ? P.A. Supposons que p^2 divise m.

$\varphi(m)$ est divisible par $p(p-1)$ et en particulier par p.
Dès lors, $\mathbb{Z}_m^*$ qui est un groupe contenant $\varphi(m)$ éléments contient un sous-groupe d'ordre p et donc aussi un élément $b \in \mathbb{Z}_m^*$ premier avec m et d'ordre p modulo m.

Lemme de Cauchy

Tout groupe commutatif fini dont l'ordre est divisible par un nombre premier p contient un élément d'ordre p.
Dans le cas général, conséquence du premier théorème de Sylow : tout groupe fini dont l'ordre est divisible par un nombre premier p contient un élément d'ordre p.

Puisque m est de Carmichael, $b^{m-1} \equiv 1 \mod m$ donc p doit diviser $m-1$.

Impossible car p divise m.

$\Leftarrow$ Supposons m est sans carré et $p-1$ divise $m-1$ pour tout facteur premier p de m.

Soient a premier avec m et p un diviseur premier de m.

Thèse : $a^{m-1} \equiv 1 \mod m$.

petit théorème de Fermat : $a^{p-1} \equiv 1 \mod p$.
Puisque $m-1$ est un multiple de $p-1$, : $a^{m-1} \equiv 1 \mod p$.
Cette congruence est OK pour tout diviseur premier de m.

Or $m = p_1 \cdots p_r$ (avec $p_i \neq p_j$, si $i \neq j$ car m est sans carré), alors pour tout $i = 1, \ldots, r$, $a^{m-1} \equiv 1 \mod p_i$.
De là, on en tire que

$$a^{m-1} \equiv 1 \mod m.$$

Corollaire

Un nombre m de Carmichael est toujours impair.

Si $p \geq 3$ est un facteur premier de m, $p-1$ divise $m-1$.
Or $p-1$ est pair donc $m-1$ l'est aussi.

Remarque

Il existe une infinité de nombres de Carmichael :
W. R. Alford, A. Granville, C. Pomerance, There Are Infinitely Many Carmichael Numbers, Ann. Math. **139**, *703–722, (1994)*.

Le nombre de nombres de Carmichael inférieurs à 10^{17} est 585355 à comparer avec $\pi(10^{17}) \sim 2,5.10^{15}$.

Pour la fonction comptant le nombre de nombres de Carmichael $\leq n$, pas de résultats analogues à $\pi(n)$. Recourir à des techniques de calcul élaborées pour les énumérer :
R.G.E. Pinch, The Carmichael numbers up to 10^{15}*, Mathematics of Computation* **61**, *381–391 (1993).*

Test de Miller-Rabin

Soit $m > 1$ un nombre impair. On pose

$$s = \max\{r \in \mathbb{N} \mid 2^r \text{ divise } m-1\} \quad \text{et} \quad d = \frac{m-1}{2^s}.$$

Théorème (admis) "à la Fermat"

Si m est un nombre premier et si $x \in \mathbb{Z}$ est premier avec m, alors l'une des deux conditions est satisfaite :

- $x^d \equiv 1 \mod m$,
- il existe $r \in \{1, \ldots, s-1\}$ tel que $x^{2^r d} \equiv -1 \mod m$.

$\Rightarrow$ Algorithme pour tester la primalité d'un entier m.

Si un entier x premier avec m est tel que aucune des deux conditions du thm n'est satisfaite, alors m est composé.

x est témoin du caractère composé de m. Sinon m est un nombre pseudo-premier fort (strong pseudoprime) pour la base x.

Proposition

Si $m \geq 3$ est impair, il y a au plus $(m-1)/4$ entiers $x < m$ premiers avec m et non témoins du caractère composé de m.

Corollaire

Si on applique k fois consécutivement le test de Miller-Rabin à un nombre composé m, la probabilité de ne pas découvrir son caractère composé est inférieure à 4^{-k}.

Exemple

$m = 561$ est un nombre de Carmichael. Si on applique le test de Miller-Rabin, on peut choisir $x = 2$. La plus grande puissance de 2 qui divise 560 est $16 = 2^4$ (car $560 = 16.35$). Ainsi, $s = 4$ et $d = 560/16 = 35$. On calcule

$$x^d = 2^{35} \mod 561 = 263.$$

Puisque le résultat n'est pas 1, on calcule $x^{2^r d}$ pour $r = 1, 2, \ldots, s-1$:

$$2^{2.35} \mod 561 = 166,\ 2^{4.35} \mod 561 = 67,\ 2^{8.35} \mod 561 = 1.$$

Par conséquent, aucune congruence ne donnant -1, on en déduit que 561 est composé et 2 en est le témoin.

Hypothèse de Riemann

Tous les zéros complexes de la fonction

$$\zeta : \mathbb{C} \to \mathbb{C} : s \mapsto \sum_{n=1}^{\infty} \frac{1}{n^s}$$

dont la partie réelle se trouve entre 0 et 1 ont leur partie réelle exactement égale à $1/2$. Par exemple, il est facile de voir que si $\operatorname{Re} s > 1$, alors $\zeta(s) \neq 0$ et que si $\operatorname{Re} s < 0$, alors les seuls zéros de la fonction ζ sont $-2, -4, -6, \ldots$ (appelés zéros triviaux). L'hypothèse de Riemann généralisée est du même type mais pour une généralisation de la fonction ζ, à savoir les L-séries de Dirichlet.

On peut montrer, en supposant l'hypothèse de Riemann généralisée vérifiée, que si m est composé, alors il existe toujours un témoin x du caractère composé de m tel que

$$x < 2(\ln m)^2.$$

Ceci permet donc d'utiliser le test de Miller-Rabin de manière déterministe en testant tous les x jusqu'à cette borne.

AGRAWAL *et al.* (2002)

Algorithme polynomial permettant de tester la primalité d'un entier.`Primes` $\in P$

RSA ET NOMBRES COMPOSÉS

RSA dans le cas où p et q ne sont pas nécessairement premiers. On suppose ici que

$$p = p_1 p_2 \quad \text{et} \quad p_1, p_2, q \text{premiers.}$$

On a $n = p.q = p_1.p_2.q$ et $\varphi(n) = (p_1 - 1)(p_2 - 1)(q - 1)$ Par contre, ne sachant pas que p n'est pas premier, la valeur de $\varphi(n)$ effectivement calculée est $\varphi'(n) = (p - 1)(q - 1)$. Soit

$$u = \text{ppcm}(p_1 - 1, p_2 - 1, q - 1).$$

Supposons que le texte clair x est premier avec n.

Par conséquent, x est premier avec p_1, p_2, q et

$$x^{p_1-1} \equiv 1 \mod p_1,\ x^{p_2-1} \equiv 1 \mod p_2,\ x^{q-1} \equiv 1 \mod q.$$

Puisque u est multiple de $p_1 - 1$, de $p_2 - 1$ et de $q - 1$, on a aussi

$$x^u \equiv 1 \mod p_1,\ x^u \equiv 1 \mod p_2,\ x^u \equiv 1 \mod q.$$

De là, p_1, p_2, q étant premiers et distincts, on en tire

$$x^u \equiv 1 \mod n.$$

Il est clair que u divise $\varphi(n)$.

Deux situations à envisager :

- soit u divise le $\varphi'(n)$ erroné,
- ou bien u ne divise pas le $\varphi'(n)$ erroné.

- Dans le premier cas, $\alpha\, u = \varphi'(n)$ et donc

$$x^{1+k\,\varphi'(n)} = x^{1+k\,\alpha\,u} \equiv x \mod n.$$

  Par conséquent, toute paire (e, d) d'exposants de chiffrement et de déchiffrement telle que $e.d \equiv 1 \mod \varphi'(n)$ convient. (En effet, les exposants ont été calculés avec le $\varphi'(n)$ erroné.)

- Dans le second cas, les choses ne se passent pas aussi bien. En général, $d_k(e_k(x)) \neq x$! Ceci est vérifié sur l'exemple suivant.

ILLUSTRATION

Soient

$$p = 391 (= 17.23) \text{ et } q = 281.$$

Ici, $n = 109871$ et le $\varphi'(n)$ calculé $= 390.280 = 109200$.

$$u = \text{ppcm}\,(16 = 2^4, 22 = 2.11, 280 = 2^3.5.7) = 2^4.5.7.11 = 6160.$$

Ici, u ne divise pas $\varphi'(n) = 109200 = 17.6160 + 4480$.

$e = 19$, on obtient l'inverse de e modulo $\varphi'(n)$, $d = 45979$.

Le texte clair $x = 8$ est bien premier avec n.
Nous devrions avoir $x^{ed} \equiv x \mod n$, mais

$$x^{ed} \mod 109871 = 95548 \neq 8.$$

Algorithmes de factorisation

- le crible d'Eratostène,
- le crible quadratique (quadratic sieve),
- des algorithmes sur les courbes elliptiques,
- le crible algébrique (number field sieve),
- l'algorithme ρ,
- la méthode $p-1$ de Pollard,
- la méthode $p+1$ de Williams,
- l'algorithme de factorisation par fractions continues,...

Le crible d'Eratostène

2	12	22	32	42	52
3	13	23	33	43	53
4	14	24	34	44	54
5	15	25	35	45	55
6	16	26	36	46	56
7	17	27	37	47	57
8	18	28	38	48	58
9	19	29	39	49	59
10	20	30	40	50	60
11	21	31	41	51	61

Le crible d'Eratostène

2	12	22	32	42	52
3	13	23	33	43	53
4	14	24	34	44	54
5	15	25	35	45	55
6	16	26	36	46	56
7	17	27	37	47	57
8	18	28	38	48	58
9	19	29	39	49	59
10	20	30	40	50	60
11	21	31	41	51	61

Le crible d'Eratostène

2	~~12~~	~~22~~	~~32~~	~~42~~	~~52~~
3	13	23	33	43	53
~~4~~	~~14~~	~~24~~	~~34~~	~~44~~	~~54~~
5	15	25	35	45	55
~~6~~	~~16~~	~~26~~	~~36~~	~~46~~	~~56~~
7	17	27	37	47	57
~~8~~	~~18~~	~~28~~	~~38~~	~~48~~	~~58~~
9	19	29	39	49	59
~~10~~	~~20~~	~~30~~	~~40~~	~~50~~	~~60~~
11	21	31	41	51	61

Le crible d'Eratostène

2	~~12~~	~~22~~	~~32~~	~~42~~	~~52~~
3	13	23	33	43	53
~~4~~	~~14~~	~~24~~	~~34~~	~~44~~	~~54~~
5	15	25	35	45	55
~~6~~	~~16~~	~~26~~	~~36~~	~~46~~	~~56~~
7	17	27	37	47	57
~~8~~	~~18~~	~~28~~	~~38~~	~~48~~	~~58~~
9	19	29	39	49	59
~~10~~	~~20~~	~~30~~	~~40~~	~~50~~	~~60~~
11	21	31	41	51	61

Le crible d'Eratostène

2	~~12~~	~~22~~	~~32~~	~~42~~	~~52~~
3	13	23	~~33~~	43	53
~~4~~	~~14~~	~~24~~	~~34~~	~~44~~	~~54~~
5	~~15~~	25	35	~~45~~	55
~~6~~	~~16~~	~~26~~	~~36~~	~~46~~	~~56~~
7	17	~~27~~	37	47	~~57~~
~~8~~	~~18~~	~~28~~	~~38~~	~~48~~	~~58~~
~~9~~	19	29	~~39~~	49	59
~~10~~	~~20~~	~~30~~	~~40~~	~~50~~	~~60~~
11	~~21~~	31	41	~~51~~	61

Le crible d'Eratostène

2	~~12~~	~~22~~	~~32~~	~~42~~	~~52~~
3	13	23	~~33~~	43	53
~~4~~	~~14~~	~~24~~	~~34~~	~~44~~	~~54~~
5	~~15~~	25	35	~~45~~	55
~~6~~	~~16~~	~~26~~	~~36~~	~~46~~	~~56~~
7	17	~~27~~	37	47	~~57~~
~~8~~	~~18~~	~~28~~	~~38~~	~~48~~	~~58~~
~~9~~	19	29	~~39~~	49	59
~~10~~	~~20~~	~~30~~	~~40~~	~~50~~	~~60~~
11	~~21~~	31	41	~~51~~	61

Le crible d'Eratostène

2	~~12~~	~~22~~	~~32~~	~~42~~	~~52~~
3	13	23	~~33~~	43	53
~~4~~	~~14~~	~~24~~	~~34~~	~~44~~	~~54~~
5	~~15~~	~~25~~	~~35~~	~~45~~	~~55~~
~~6~~	~~16~~	~~26~~	~~36~~	~~46~~	~~56~~
7	17	~~27~~	37	47	~~57~~
~~8~~	~~18~~	~~28~~	~~38~~	~~48~~	~~58~~
~~9~~	19	29	~~39~~	49	59
~~10~~	~~20~~	~~30~~	~~40~~	~~50~~	~~60~~
11	~~21~~	31	41	~~51~~	61

Le crible d'Eratostène

2	~~12~~	~~22~~	~~32~~	~~42~~	~~52~~
3	13	23	~~33~~	43	53
~~4~~	~~14~~	~~24~~	~~34~~	~~44~~	~~54~~
5	~~15~~	~~25~~	~~35~~	~~45~~	~~55~~
~~6~~	~~16~~	~~26~~	~~36~~	~~46~~	~~56~~
7	17	~~27~~	37	47	~~57~~
~~8~~	~~18~~	~~28~~	~~38~~	~~48~~	~~58~~
~~9~~	19	29	~~39~~	49	59
~~10~~	~~20~~	~~30~~	~~40~~	~~50~~	~~60~~
11	~~21~~	31	41	~~51~~	61

Le crible d'Eratostène

2	~~12~~	~~22~~	~~32~~	~~42~~	~~52~~
3	13	23	~~33~~	43	53
~~4~~	~~14~~	~~24~~	~~34~~	~~44~~	~~54~~
5	~~15~~	~~25~~	~~35~~	~~45~~	~~55~~
~~6~~	~~16~~	~~26~~	~~36~~	~~46~~	~~56~~
7	17	~~27~~	37	47	~~57~~
~~8~~	~~18~~	~~28~~	~~38~~	~~48~~	~~58~~
~~9~~	19	29	~~39~~	~~49~~	59
~~10~~	~~20~~	~~30~~	~~40~~	~~50~~	~~60~~
11	~~21~~	31	41	~~51~~	61

Le crible d'Eratostène

2	~~12~~	~~22~~	~~32~~	~~42~~	~~52~~
3	13	23	~~33~~	43	53
~~4~~	~~14~~	~~24~~	~~34~~	~~44~~	~~54~~
5	~~15~~	~~25~~	~~35~~	~~45~~	~~55~~
~~6~~	~~16~~	~~26~~	~~36~~	~~46~~	~~56~~
7	17	~~27~~	37	47	~~57~~
~~8~~	~~18~~	~~28~~	~~38~~	~~48~~	~~58~~
~~9~~	19	29	~~39~~	~~49~~	59
~~10~~	~~20~~	~~30~~	~~40~~	~~50~~	~~60~~
11	~~21~~	31	41	~~51~~	61

Le crible d'Eratostène

Imaginons un ordinateur capable de réaliser 10^9 divisions par seconde ! Si n est de l'ordre de 2^{1024}, $\sqrt{n} \sim 10^{150}$. Il faudrait donc au crible d'Eratostème près de 10^{140} secondes ce qui dépasse de très loin l'âge de l'univers !

Méthode $p-1$ de Pollard (1974)

Factoriser n et on se fixe une borne B

Algorithme de Pollard

Choisir aléatoirement $g \in \{2, \ldots, n-1\}$.
Calculer $t = \mathrm{pgcd}(g, n)$.
Si $t > 1$, alors on a trouvé un facteur de n.
Sinon, calculer $a = g^{B!} \mod n$.
Calucler $d = \mathrm{pgcd}(a-1, n)$.
Si $d > 1$, d est un facteur.
Sinon, facteur non trouvé, recommencer avec $B >$.

Soit p un facteur premier de n tq toute puissance r^α d'un nombre premier r, divisant $p-1$, soit $\leq B$.

$$p-1 = r_1^{\alpha_1} \cdots r_t^{\alpha_t} \text{ avec } r_i^{\alpha_i} \leq B, \ \forall i.$$

Si B est "suffisamment petit", il est facile de déterminer un multiple de $p-1$ sans pour autant connaître de $p-1$.

Puisque $a \equiv g^{B!} \mod n$, on a $a \equiv g^{B!} \mod p$ car $p|n$. Par le petit théorème de Fermat : $g^{p-1} \equiv 1 \mod p$
(si g est premier avec p, ce que nous supposons vu l'algorithme).

Puisque tout facteur r^α de $p-1$ est $\leq B$, on a

$$p-1 \text{ divise } B!$$

car $r_i \neq r_j$ et chaque $r_i^{\alpha_i} \leq B$.

De là,

$$a \equiv g^{B!} = g^{m\,(p-1)} = (g^{p-1})^m \equiv 1 \mod p$$

et $a - 1$ est un multiple de p. Par conséquent, il ne reste plus qu'à calculer le pgcd de n et de $a - 1$.

On retrouve donc d'une certaine façon la mise en garde **3.** concernant le choix de p et q dans l'élaboration du RSA.

Illustration

Considérons l'entier (composé) $n = 15770708441$.
Avec $g = 2$ et B fixé à 200, on calcule

$$g^{200!} \mod n = 6094850739.$$

Ensuite, on calcule le pgcd de $6094850739 - 1$ et de n et on trouve

$$135979$$

qui est donc un facteur de n.

Illustration

En fait, n est le produit des deux nombres premiers suivants :

$$p = 115979 \ (p - 1 = 2.103.563)$$

$$q = 135979 \ (q - 1 = 2.3.131.173).$$

Une borne $B \geq 173$ permettra la factorisation de n.

On pourra observer qu'une borne $B \geq 563$ ne donnera plus de résultat, car dans ce cas, a sera congru à 1 modulo p et q donc aussi modulo n. De toute façon, en pratique, on considère les bornes les plus petites possibles.

Remarque

Cet algorithme fonctionne en un temps polynomial en n à condition que la borne convenanble B soit en $\mathcal{O}((\log n)^i)$.

Cependant, pour un grand nombre n arbitraire, il est probable que la borne B doive augmenter jusque $\sqrt{n}$ et dans ce cas, la méthode $p-1$ n'est pas plus rapide que le crible d'Eratostène.

- 1998, des nombres $\sim 10^{70}$ pouvaient être factorisés en 10 heures sur une station de travail.
- A cette même époque, il fallait un an pour factoriser un nombre de 100 chiffres décimaux sur une station.
- 1999, un nombre de 155 chiffres a été factorisé en plus de 5 mois par 292 ordinateurs en réseau. Des calculs préalables à la factorisation ont pris près de 4 mois à des super-ordinateurs CRAY.
- Décembre 2003, le "RSA Challenge number" `RSA-576` de 576 bits (174 chiffres décimaux) a été factorisé `http ://www.rsasecurity.com/`

- Le “RSA Challenge number” RSA-640 de 640 bits suivant

  3107418240490043721350750035888567930037346022842727545720161948823206440518081504556346829671723286782437916272838033415471073108501919548529007337724822783525742386454014691736602477652346609

  factorisé le 5 novembre 2005 (193 chiffres décimaux), 20000$. 4 mois de calculs en réseau, représente plus de 30 ans de calcul pour un seul processeur Opteron cadencé à 2,2 Ghz.
- A titre indicatif, on propose 100000$ pour le challenge number RSA-1024 . . .

- Le “RSA Challenge number” RSA-640 de 640 bits suivant

  3107418240490043721350750035888567930037346022842727545720161948823206440518081504556346829671723286782437916272838033415471073108501919548529007337724822783525742386454014691736602477652346609

  factorisé le 5 novembre 2005 (193 chiffres décimaux), 20000$. 4 mois de calculs en réseau, représente plus de 30 ans de calcul pour un seul processeur Opteron cadencé à 2,2 Ghz.
- A titre indicatif, on propose 100000$ pour le challenge number RSA-1024 . . .

Let us quote

"*Using a minimal key length of* 1024 *bits, it is guaranteed that RSA can be considered safe for the near future as long as there will be no fundamental advance in the factoring of large numbers.*"

Let us quote

"Clearly, the factoring of a challenge-number of specific length does not mean that the RSA cryptosystem is "broken." It does not even mean, necessarily, that keys of the same length as the factored challenge number must be discarded. It simply gives us an idea of the amount of work required to factor a modulus of a given size. This can be translated into an estimate of the cost of breaking a particular RSA key pair.

Suppose, for example, that in the year 2010 a factorization of RSA-768 is announced that requires 6 months of effort on 100,000 workstations. In this hypothetical situation, would all 768-bit RSA keys need to be replaced? The answer is no. If the data being protected needs security for significantly less than six months, and its value is considerably less than the cost of running 100,000 workstations for that period, then 768-bit keys may continue to be used."

Logarithme discret

Soit G un groupe multiplicatif cyclique contenant n éléments et γ un générateur de G.

Définition

Soit $x \in G$. Le logarithme discret de x est le plus petit d tq

$$x = \gamma^d.$$

$\mathrm{dlog}_\gamma : G \to \{1, \ldots, |G|\}$, $\mathrm{dlog}\, x = d$ ou $\mathrm{dlog}_\gamma\, x = d$

Exemple

Dans $\mathbb{Z}_{17}^*$, 3 est générateur,

i	1	2	3	4	5	6	7	8	9	10	11	12	13	$\cdots$
3^i	3	9	10	13	5	15	11	16	14	8	7	4	12	$\cdots$

$$\mathrm{dlog}_3\, 15 = 6 \text{ et } \mathrm{dlog}_3\, 14 = 9.$$

Fonction à sens unique

Si $G = \mathbb{Z}_p^*$ où p est un grand nombre premier, alors $\mathbb{Z}_p^*$ est cyclique. Si on se donne un générateur γ et un élément $x \in \mathbb{Z}_p^*$, le calcul du logarithme discret est conjecturé être difficile.

Par contre, exponentiation (modulaire) est “facile”.

- l’algorithme “baby-step giant-step” de Shanks,
- la ρ-méthode de Pollard,
- l’algorithme de Pohlig-Hellman,
- le calcul d’indices,. . .

Aucun algorithme connu à ce jour ne donne de résultats satisfaisant pour des nombres premiers p arbitraires suffisamment grands.

Les complexités des algorithmes de factorisation de grands nombres et de recherche de logarithme discret sont proches.

Remarque

Si $G = (\mathbb{Z}_p^*, \cdot)$, alors $\mathbb{Z}_p^*$ est cyclique d'ordre $p-1$ donc isomorphe au groupe additif $(\mathbb{Z}_{p-1}, +)$. Il existe une bijection

$$\psi : (\mathbb{Z}_p^*, \cdot) \to (\mathbb{Z}_{p-1}, +)$$

tq. $\psi(xy \mod p) = \psi(x) + \psi(y) \mod p-1$.
On en tire que $\psi(\gamma^a \mod p) = a\,\psi(\gamma) \mod p-1$. Donc

$$\begin{aligned} x \equiv \gamma^a \mod p \quad &\Leftrightarrow \quad \psi(x) \equiv a\,\psi(\gamma) \mod p-1 \\ &\Leftrightarrow \quad \mathrm{dlog}_\gamma\, x = a \equiv \psi(x)(\psi(\gamma))^{-1} \mod p-1. \end{aligned}$$

Si on dispose d'une méthode efficace pour calculer ψ, on obtient alors un algorithme efficace pour calculer le logarithme discret dans $\mathbb{Z}_p^*$ car les calculs dans $(\mathbb{Z}_{p-1}, +)$ sont simples à effectuer.

Remarque

- On ne connaît pas de méthode générale pour déterminer ψ pour un nombre premier arbitraire p.
- La détermination de ψ est aussi difficile que la détermination du logarithme discret.
- Si pour un groupe G quelconque, l'isomorphisme est facile à obtenir, il vaut mieux ne pas utiliser un tel groupe pour construire un cryptosystème.
- Pour des groupes construits sur des courbes elliptiques ou sur le groupe multiplicatif $\mathbb{F}_{p^f}^*$, on ne dispose pas de méthode générale de recherche de ψ.

Protocole d'échange des clés de Diffie-Hellman

Bob et Alice choisissent

- un grand nombre premier p
- une racine primitive γ modulo p.

Alice choisit un exposant a (secret) et envoie à Bob

$$A = \gamma^a \mod p.$$

Bob choisit un exposant b et envoie à Alice

$$B = \gamma^b \mod p.$$

Alice (connaissant a) peut calculer $B^a = \gamma^{ab} \mod p$
Bob (connaissant b) peut calculer $A^b = \gamma^{ab} \mod p$

Cette valeur commune leur sert à présent de clé secrète commune.

Si Oscar espionne les échanges entre Alice et Bob, alors il a sa disposition les éléments suivants :

$$p,\ \gamma, A \text{ et } B.$$

Si le problème du logarithme discret est difficile, ne connaissant ni a ni b, il n'est pas en mesure de retrouver la clé secrète $B^a = A^b \mod p$.

Attaque “Man in the middle”

Pour se prémunir d’une telle attaque, il est dès lors nécessaire de recourir à un procédé de signature.

CRYPTOSYSTÈME D'ELGAMAL

Nous l'envisageons dans $\mathbb{Z}_p^*$ mais s'adapte à d'autres groupes ($\mathbb{F}_{p^f}^*$ ou groupes sur les courbes elliptiques).

Pour assurer la sécurité du cryptosystème, prendre $p \geq 2^{768}$.

On choisit un grand nombre premier p et $\gamma \in \mathbb{Z}_p^*$.

préférable mais pas nécessaire que γ soit une racine primitive modulo p.

TIRER γ AU HASARD

Dans $\mathbb{F}_p^*$, le nombre de générateurs est égal à $\varphi(p-1)$
J. Rosser and L. Schoenfeld, Approximate formulas for some functions of prime numbers, Illinois J. of Math. **6***, 69–94, (1962).*

$$\forall n \geq 5,\ \varphi(n) \geq \frac{n}{\log\log n}.$$

En tirant un élément γ au hasard dans $\mathbb{Z}_p^*$, on a de “bonnes chances” qu’il s’agisse d’une racine primitive puisque la proportion de tels éléments dans $\mathbb{Z}_p^*$ est d’au moins $1/\log\log(p-1)$.

Tirer γ au hasard

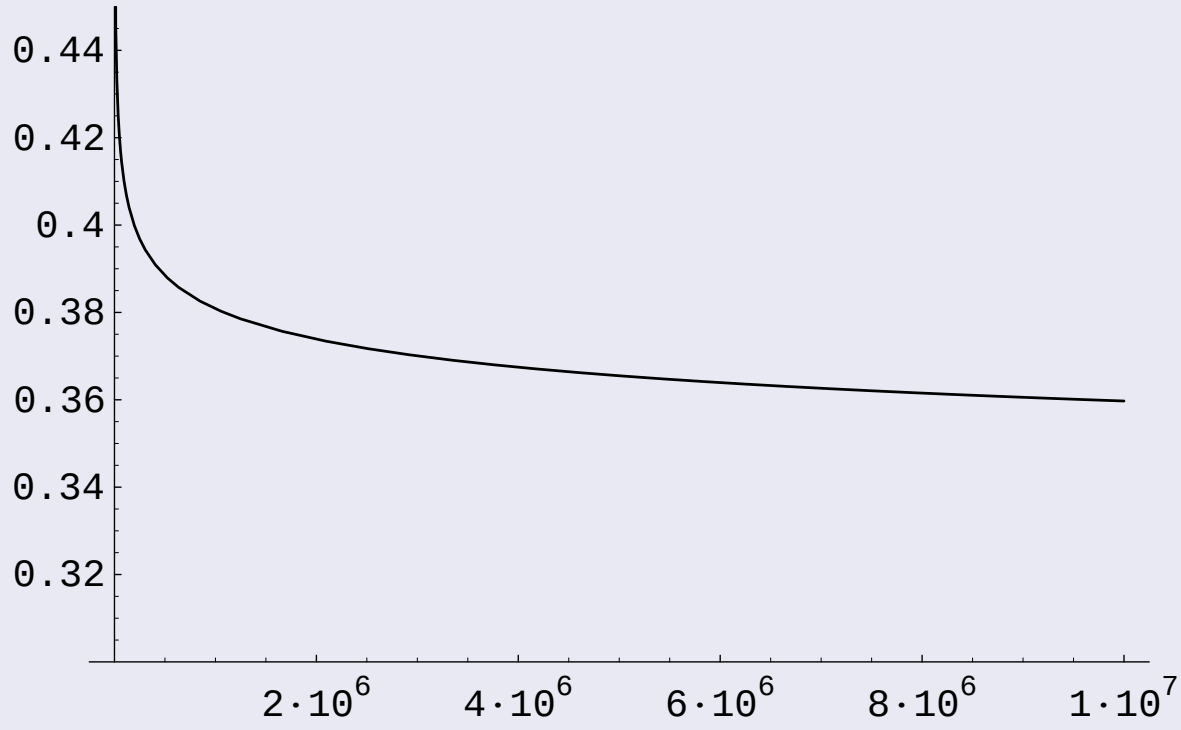


Fig.: Graphique de $1/\log\log x$ pour $x \leq 10^7$.

$1/\log\log 2^{768} \sim 0,1593$. Cette borne est la plus défavorable. Si $p-1 = 2.q$ avec q un nombre premier, alors $\varphi(p-1) = q-1$ et la probabilité de tirer au hasard une racine primitive est dès lors proche de $1/2$.

L'ensemble des textes clairs est $\mathcal{P} = \mathbb{Z}_p$ (on emploie, si nécessaire, des conventions de codage).

Bob choisit aléatoirement un exposant $0 < d < p - 1$ (secret) et calcule

$$e = \gamma^d \mod p.$$

Bob publie

$$k = (p, \gamma, e).$$

On appelle parfois e la clé de Diffie-Hellman de Bob.

Si Alice veut envoyer un message x à Bob, elle choisit un nombre aléatoire $0 < b < p - 1$ et envoie à Bob le couple

$$(\gamma^b \mod p,\ e^b x \mod p).$$

Remarque $e^b = (\gamma^d)^b = (\gamma^b)^d$.

Pour le déchiffrement, Bob reçoit $(\gamma^b \mod p,\ e^b x \mod p)$ et doit retrouver x.

Pour éviter de calculer $(e^b)^{-1} \mod p$ (ce qui, par ailleurs, est supposé difficile puisqu'il faudrait calculer le logarithme discret $\mathrm{dlog}_\gamma \gamma^b$ pour retrouver b),

Bob connaissant d, connaît aussi $p-1-d$ et il lui suffit alors de calculer $(\gamma^b)^{p-1-d} e^b x \mod p$ pour retrouver x. En effet,

$$(\gamma^b)^{p-1-d} e^b x \equiv (\gamma^{p-1})^b (\gamma^b)^{-d} e^b x$$

$$\equiv (\gamma^{p-1})^b (\gamma^b)^{-d} (\gamma^b)^d x \equiv x \mod p.$$

Que γ soit ou non une racine primitive modulo p, on a toujours $\gamma^{p-1} \equiv 1 \mod p$ car l'ordre d'un élément de $\mathbb{Z}_p^*$ divisant l'ordre du groupe, $p-1$ est un multiple de l'ordre de γ.

Calculs nécessaires

Du même type que pour le RSA, mais le RSA est bien plus lent.

Il s'agit simplement d'exponentiations modulaires. Le déchiffrement nécéssite une exponentiation et le chiffrement en nécéssite deux.

Cependant, les calculs de $\gamma^b \mod p$ et $e^b \mod p$ peuvent être réalisés une fois pour toutes par Alice (elle choisit un seul b pour tous ces textes clairs x) et donc être supposés précalculés.

Par conséquent, le chiffrement d'ElGamal ne nécéssite en fin de compte qu'une multiplication modulo p ce qui est de loin bien plus efficace que le RSA.

```
p = Prime[123456789]
```

2543568463

```
gamma = Prime[12345678]
```

224284387

```
d = 12345678
```

12345678

```
e = PowerMod[gamma, d, p]
```

831108609

```
texteclair =
 "une douleur foudroyante lui traversa la tete,une douleur comme il n’en avait
   encore jamais ressenti. c’etait comme si sa cicatrice avait soudain pris feu."
```

une douleur foudroyante lui traversa la tete,une douleur comme il n’en avait
encore jamais ressenti. c’etait comme si sa cicatrice avait soudain pris feu.

```
N[Log[32, p]]
```

6.24884

```
codetxclair = code[texteclair]
```

```
{21, 14, 5, 0, 4, 15, 21, 12, 5, 21, 18, 0, 6, 15, 21, 4, 18, 15, 25, 1, 14, 20, 5, 0, 12, 21,
 9, 0, 20, 18, 1, 22, 5, 18, 19, 1, 0, 12, 1, 0, 20, 5, 20, 5, 27, 21, 14, 5, 0, 4, 15,
 21, 12, 5, 21, 18, 0, 3, 15, 13, 13, 5, 0, 9, 12, 0, 14, 29, 5, 14, 0, 1, 22, 1, 9, 20,
 0, 5, 14, 3, 15, 18, 5, 0, 10, 1, 13, 1, 9, 19, 0, 18, 5, 19, 19, 5, 14, 20, 9, 28, 0, 3,
 29, 5, 20, 1, 9, 20, 0, 3, 15, 13, 13, 5, 0, 19, 9, 0, 19, 1, 0, 3, 9, 3, 1, 20, 18, 9,
 3, 5, 0, 1, 22, 1, 9, 20, 0, 19, 15, 21, 4, 1, 9, 14, 0, 16, 18, 9, 19, 0, 6, 5, 21, 28}
```

```
Length[codetxclair]
```

```
154
```

```
codetxclair = ajout[codetxclair, 6]
```

```
{21, 14, 5, 0, 4, 15, 21, 12, 5, 21, 18, 0, 6, 15, 21, 4, 18, 15, 25, 1, 14, 20, 5, 0, 12, 21, 9,
 0, 20, 18, 1, 22, 5, 18, 19, 1, 0, 12, 1, 0, 20, 5, 20, 5, 27, 21, 14, 5, 0, 4, 15, 21,
 12, 5, 21, 18, 0, 3, 15, 13, 13, 5, 0, 9, 12, 0, 14, 29, 5, 14, 0, 1, 22, 1, 9, 20, 0, 5,
 14, 3, 15, 18, 5, 0, 10, 1, 13, 1, 9, 19, 0, 18, 5, 19, 19, 5, 14, 20, 9, 28, 0, 3, 29,
 5, 20, 1, 9, 20, 0, 3, 15, 13, 13, 5, 0, 19, 9, 0, 19, 1, 0, 3, 9, 3, 1, 20, 18, 9, 3, 5,
 0, 1, 22, 1, 9, 20, 0, 19, 15, 21, 4, 1, 9, 14, 0, 16, 18, 9, 19, 0, 6, 5, 21, 28, 0, 0}
```

```
liste = codebloc[codetxclair, 6]
```

```
{719487119, 717411904, 217748047, 840388768, 424968850, 56805985,
 12616325, 677238213, 4707717, 723521005, 441460096, 500348929,
 739561477, 473417888, 337020211, 19058277, 491057155, 978978100, 3650981,
 20218465, 3443764, 613520385, 739561491, 525468974, 17376864, 207286272}
```

```
b = Random[Integer, {1, p - 1}]
```

```
2077626273
```

```
PowerMod[gamma, b, p]
```

```
2174065976
```

```
temp = PowerMod[e, b, p]
```

```
401303819
```

```
codetxchiffre = Mod[temp * liste, p]
```

```
{1224703372, 1813996580, 879587936, 965469907, 1572901588, 1078313219, 1291506749,
 397171217, 2052820288, 1630733064, 162334271, 1346975257, 1481548401,
 31283547, 452298537, 2531435220, 2538016017, 2419180759, 1225202253, 373714894,
 1135501389, 1561257229, 2012664941, 1607195455, 371515150, 1134385436}
```

```
decodebloc[codetxchiffre, 7]
```

```
ado:?llava:yad zfz.s  .xwyvsan.advta dkphcafouua' kzxvpqa'ewcj
  apsf xh dzzaq?ahdrmpyald'hsq  'zvj, mokayibknehvtbktnchqbhccqfwadpnfrm
  kdl,:naaz.xbmanp'yhma,?mucmao.wvy? kbiwxnaayzvh.
```

```
temp = PowerMod[2174065976, (p - 1 - d), p]
```

```
438045370
```

```
Mod[temp * codetxchiffre, p]
```

```
{719487119, 717411904, 217748047, 840388768, 424968850, 56805985,
 12616325, 677238213, 4707717, 723521005, 441460096, 500348929,
 739561477, 473417888, 337020211, 19058277, 491057155, 978978100, 3650981,
 20218465, 3443764, 613520385, 739561491, 525468974, 17376864, 207286272}
```