Informatique MP
Eléments d'arithmétique pour la cryptographie

Antoine MOTEAU
antoine.moteau@wanadoo.fr

Dernière rédaction : 27 Juillet 2003

. . . /arithcri.tex (2003)
. . . /arithcri.tex Compilé le samedi 13 avril 2019 à 11h 36m 48s avec PDFLaTeX.
Compilable avec LaTeX, PDFLaTeX, LuaLaTeX, XeLaTeX.

Document à compléter au fil du temps . . . :

- Autres algorithmes
- Démonstrations,
- Plus de références

Antoine MOTEAU – Lycée POTHIER – Option Informatique MP – Maj le 13 avril 2019
Document d'information pour le TP RSA.

Eléments d'arithmétique pour la Cryptographie

Introduction

Les méthodes de cryptographie sont basées, pour l'essentiel, sur la difficulté à effectuer certains calculs avec de grands nombres entiers (couramment plus de 100 chiffres décimaux) :

- factorisation d'un entier n,
- calculs dans $\frac{\mathbb{Z}}{n\mathbb{Z}}$ (calculs modulo l'entier n)
  - recherche d'un inverse dans $\frac{\mathbb{Z}}{n\mathbb{Z}}$
  - recherche d'un logarithme discret dans $\frac{\mathbb{Z}}{n\mathbb{Z}}$ lorsque n est premier.

  (résoudre, dans $\frac{\mathbb{Z}}{n\mathbb{Z}}$, l'équation d'inconnue x : $a^x = b \mod n$)

Lorsque l'on a affaire à des nombres gigantesques et à des calculs utilisant les particularités de ces nombres, seul ceux qui connaissent ces particularités sont capables de réaliser de tels calculs.

L'objectif de sécurité des méthodes de cryptographie est de faire en sorte que (du moins pendant un certain temps), la recherche à priori des particularités d'un nombre gigantesque bien choisi soit irréalisable en un temps fini.

Certains calculs, qui se réalisent simplement en connaissant la décomposition d'un nombre en ses facteurs premiers, sont impossibles à réaliser autrement. C'est ce genre de particularité qui explique l'importance accordée à la recherche de nombres entiers de plus en plus grands, de plus en plus difficiles à factoriser, donc à la recherche de nombres premiers les plus grands possibles, ainsi qu'à l'évaluation du temps nécessaire à la factorisation de nombres gigantesques :

- En 1990, factorisation d'un nombre de 69 chiffres qui était un facteur non premier de $2^{512} - 1$ = 1340780792 9942597099 5740249982 0584612747 9365820592 3933777235 6144372176 4030073546 9768018742 9816690342 7690031858 1864860508 5375388281 1946569946 4336490060 84095 (155 chiffres)
- En 1994, factorisation d'un nombre de 129 chiffres (...) (temps de calcul : 5000 MIPS-année).
- En 1999, Ref [NOBS 28/07/99]. Nayan Hajratwala (simple amateur ?) a été le premier à découvrir un nombre premier de plus d'un million de chiffres. Trois semaines de calculs (avec quels moyens ?) ont été nécessaires pour prouver que $2^{6972593} - 1$ est premier.
- 2008, 2012 ...
- En 2018 (le 3 janvier 2018), le plus grand nombre premier connu est le nombre premier de Mersenne $2^{77232917}-1$, qui comporte plus de 23 millions de chiffres en écriture décimale.
- ...

La représentation des grands nombres entiers

Pour représenter exactement des grands nombres entiers, il est nécessaire de faire appel à une bibliothèque spécialisée dans la gestion de structures adaptées à de telles représentations.

En `CaML`, la bibliothèque `big_int` offre comme ressources

- un type `big_int` de représentation exacte d'entiers de taille quelconque
- des fonctions de calculs numérique adaptées à ce type

et peut être complétée par quelques fonctions utiles.

Remarque. Le type `big_int`, de la bibliothèque `big_int` de `CaML`, est un type "abstrait" (dont la structure n'est pas communiquée), ce qui oblige à n'utiliser que les fonctions (mal) documentées et déjà présentes dans la bibliothèque `big_int` comme constituants de nouvelles fonctions.

1 Quelques "rappels", "évocations" mathématiques ...

Sans démonstrations. Certaines notions ne sont pas au programme de mathématiques de spé ....

- Structures algébriques usuelles,
  - **Groupes.**

    Exemples : $(\mathbb{Z},+)$, $\left(\frac{\mathbb{Z}}{n\mathbb{Z}},+\right)$ et, lorsque n est premier, $\left(\left(\frac{\mathbb{Z}}{n\mathbb{Z}}\right)^*,\times\right)$
  - **Anneaux , idéaux.**

    Exemples : Anneau $(\mathbb{Z},+,\times)$, Idéal $n\mathbb{Z}$, Anneau quotient $\left(\frac{\mathbb{Z}}{n\mathbb{Z}},+,\times\right)$.
  - **Corps.**

    Exemples : $(\mathbb{Q},+,\times)$, $(\mathbb{R},+,\times)$, $(\mathbb{C},+,\times)$, et, lorsque n est premier, $\left(\frac{\mathbb{Z}}{n\mathbb{Z}},+,\times\right)$.
- **Théorème**

  Si $(G,\times)$ est un groupe multiplicatif fini d'ordre n, l'ordre de tout élément de G divise n.
- **Exponentiation rapide (algorithme)**

```
let rec exp_rap a = function   (* version récursive, pour le type entier de base *)
    0 -> 1
  | p -> let u = exp_rap a p/2 in if p mod 2 = 0 then u * u else u * u * a
;;
(* exp_rap : int -> int -> int = <fun> *)
```

  Remarques.
  - On peut aussi écrire une version itérative pour l'exponentiation rapide.
  - Dans la suite on prendra la multiplication "modulo n" pour le calcul de $a^p \mod n$.

  La complexité (temporelle) de `(exp_rap a p)` est un $O(\log_2 p)$ multiplications d'entiers.
- **Division euclidienne dans $\mathbb{Z}$**

  ...
- **Congruence modulo n dans $\mathbb{Z}$, ensemble quotient $\frac{\mathbb{Z}}{n\mathbb{Z}}$, anneau quotient $\left(\frac{\mathbb{Z}}{n\mathbb{Z}},+,\times\right)$**

  ...
- **Calcul du pgcd de deux entiers : algorithme d'Euclide**

```
let rec pgcd x y =                          (* version récursive *)
   if y = 0 then x else pgcd y (x mod y)
;;

let pgcd x y =                              (* version itérative *)
  let x1 = ref x and y1 = ref y in
  while !y1 <> 0 do let u = !y1 in y1 := !x1 mod !y1; x1 := u done;
  !x1
;;
(* pgcd : int -> int -> int = <fun> *)
```

  La complexité temporelle est donnée par le théorème de Lamé :

  Théorème de Lamé (1845) (complexité de l'algorithme d'Euclide)

  Si u et v sont des entiers de l'intervalle $[0\cdots N[$, le nombre de divisions (mod) dans l'algorithme d'Euclide appliqué à u et v est au plus $\log_\phi\left(\sqrt{5}\,N\right)-2$ (avec $\phi=\frac{1+\sqrt{5}}{2}$).

- **Théorème de Bezout, algorithme d'Euclide étendu**

  1. **Théorème de Bezout**

     Pour $x, y \in \mathbb{N}$, il existe u et v éléments de $\mathbb{Z}$ tels que $ux+vy = pgcd(x,y)$.

  2. **Analyse d'un algorithme pour obtenir la décomposition de Bezout** :

     Objectif : déterminer $w = pgcd(x,y)$ et des entiers relatifs u et v tels que $ux+vy = w$.

     - Si $y = 0$, on a $pgcd(x,y) = x$ et $1x+0y = x$ d'où $(u,v,w) = (1,0,x)$ convient.
     - Sinon, on a $x = qy+r$ avec $0 \leqslant r < y$, et $w = pgcd(x,y) = pgcd(y,r)$.
       Alors, avec u' et v' tels que $u'y+v'r = pgcd(y,r)$ $(= w)$, on a :

       $w = u'y+v'r = u'y+v'(x-qy) = v'x+(u'-v'q)y = ux+vy$ avec $\begin{cases} u = v' \\ v = u'-v'q \end{cases}$

  3. **Algorithme récursif** :

```
let rec bezout x = function                (* version récursive *)
    0 -> 1,0,x
  | y -> let u',v',w' = bezout y (x mod y)
         in v', u' - v' * (x quo y), w'
;;
(* bezout : int -> int -> int * int * int = <fun> *)
```

  4. **Algorithme itératif** :

```
1 : U = (u0, u1, u2) <- (1, 0, x)  et  V = (v0, v1, v2) <- (0, 1, y)
2 : Tant que  v2  <>  0,  faire (U, V) <- (V, U - q * V) avec q = u2 div v2
3 : Renvoyer U
```

     Eléments pour la preuve de la version itérative :

     * invariants (relations) : $\begin{cases} u_0 x + u_1 y = u_2, \\ v_0 x + v_1 y = v_2, \\ pgcd(u_2,v_2) = pgcd(x,y) \end{cases}$
     * décroissance : quand $v_2 \neq 0$, avec $u_2 = v_2 q + r$ où $0 \leqslant r < v_2$, alors
       - la prochaine valeur de v_2 est r, ce qui prouve que la suite des valeurs de v_2 est positive, strictement décroissante.
       - Lorsque $r = 0$, $v_2 = pgcd(u_2,v_2)$ et U prend comme dernière valeur (v_0,v_1,v_2) qui vérifie $v_0x+v_1y = v_2 = pgcd(u_2,v_2) = pgcd(x,y)$

- **Inverse modulo n, division modulo n** (application du théorème de Bezout)

  Pour $n \in \mathbb{N}^*$, soit $x \in \mathbb{N}^*$ tels que $x \wedge n = 1$ $(pgcd(x,n) = 1$; n et x premiers entre eux).
  Alors $x \mod n$ est inversible dans $\left(\dfrac{\mathbb{Z}}{n\mathbb{Z}}, \times\right)$,

  - l'inverse modulo n de x est l'entier k tel que $0 < k < n$ et $kx \equiv 1 \mod n$.
  - le quotient modulo n de $y \in \mathbb{N}$ par x est l'entier q tel que $0 \leqslant q < n$ et $y = xq \mod n$.

  Avec $x \wedge n = 1$, en prenant u et v dans $\mathbb{Z}$ tels que $ux+vn = 1$, on a $ux \equiv 1 \mod n$.
  On pose $a = u \mod n$ et on prend $k = a$ si $a > 0$ et $k = a+n$ sinon, ce qui donne :

  - $kx \equiv 1 \mod n$, avec $0 < k < n$
  - avec $q = ky \mod n$, on a $xq \equiv (xk)y \equiv y \mod n$, avec $0 \leqslant q < n$

- **Théorème** (Euclide) :

  L'ensemble des nombres premiers est infini.

- **Théorème des nombres premiers**

  Le nombre $\pi(n)$ des nombres premiers inférieurs à n vérifie $\pi(n) \underset{n\to+\infty}{\sim} \dfrac{n}{\log n}$.

  d'où $\dfrac{\pi(n)}{n} \xrightarrow[n\to+\infty]{} 0$.

- **Théorème**

Pour $n \in \mathbb{N}^*$, $n > 1$; l'anneau $\left(\dfrac{\mathbb{Z}}{n\mathbb{Z}}, +, \times\right)$ est un corps si et seulement si n est premier.

- **Théorème des restes Chinois**

$m_1, \cdots, m_k$ étant des entiers supérieurs à 2 et deux à deux premiers entre eux,
le système des k équations, d'inconnue x,

$$\forall\, i = 1 \cdots k,\ x \equiv a_i \mod m_i$$

admet une unique solution modulo $M = m_1 \times \cdots \times m_k$, qui est

$$x = \left(\sum_{i=1}^{k} a_i\, M_i\, y_i\right) \mod M \qquad \text{où} \quad \forall\, i = 1 \cdots k,\ M_i = \frac{M}{m_i} \text{ et } \left(y_i = M_i^{-1} \mod m_i\right)$$

- **Indicateur d'Euler $\phi(n)$**

Pour $n \in \mathbb{N}$, $n \geqslant 2$, $\phi(n)$ est le nombre des entiers de $[\![1..n-1]\!]$, premiers avec n.

- lorsque n est premier, $\phi(n) = n - 1$,
- lorsque $n = \prod_{i=1}^{k} p_i^{e_i}$, où les p_i sont premiers et distincts, $\phi(n) = \prod_{i=1}^{k} p_i^{e_i - 1}\,(p_i - 1)$.

- **Groupe** $(\mathbb{Z}_n^*, \times)$

On note $\mathbb{Z}_n^*$ l'ensemble des éléments de $\mathbb{Z}_n = \dfrac{\mathbb{Z}}{n\mathbb{Z}}$, inversibles modulo n.

- $(\mathbb{Z}_n^*, \times)$ est un groupe multiplicatif d'ordre $\phi(n)$ (à $\phi(n)$ éléments).
- Lorsque n est premier, $(\mathbb{Z}_n^*, \times)$ est un groupe cyclique, d'ordre $n-1$.
- Un élément a de $\mathbb{Z}_n^*$ est dit **primitif** s'il est générateur de $\mathbb{Z}_n^*$.
- Il y a $\phi(p-1)$ éléments primitifs dans $\mathbb{Z}_n^*$.

Remarque. Lorsque n est premier, $\mathbb{Z}_n^* = \left(\dfrac{\mathbb{Z}}{n\mathbb{Z}}\right)^* = \left(\dfrac{\mathbb{Z}}{n\mathbb{Z}}\right) \setminus \{0\}$ et s'identifie à $\{1, 2, \ldots, n-1\}$.

- **Théorème (petit théorème de Fermat)**

Soit p un entier premier.
Si a est un entier non divisible par p, alors $a^{p-1} - 1$ est divisible par p (ou $a^p \equiv a \mod p$)

- **Théorème de Fermat-Wiles**

Soit n, un entier au moins égal à trois. Il n'existe pas de nombres entiers non tous nuls (ni même d'ailleurs de rationnels) vérifiant l'équation $x^n + y^n = z^n$.

Remarque. Par contre, pour $n = 2$, il y a une infinité de solutions, par exemple $5^2 = 3^2 + 4^2$
ou $13^2 = 5^2 + 12^2$, ou encore $29^2 = 19^2 + 20^2$, ...

Ce théorème (dernier théorème de Fermat ou grand théorème de Fermat), du mathématicien français Pierre de Fermat, magistrat au parlement de Toulouse, a été écrit sous la forme d'une courte annotation en marge d'un livre de mathématiques publié en 1641 : "Il n'est pas possible de décomposer un cube en somme de deux cubes, une puissance quatrième en somme de deux puissances quatrièmes et généralement aucune puissance d'exposant supérieur à 2 en deux puissances de même exposant", accompagnée du commentaire "J'ai trouvé une merveilleuse démonstration de cette proposition, mais je ne peux l'écrire dans cette marge car elle est trop longue".

En fait, le théorème, devenu l'un des théorèmes les plus célèbres des mathématiques, n'a été démontré qu'en 1994, par Andrew Wiles de l'Université de Princeton, après plus de 8 ans de travail !

Prix Whitehead (1988), prix Schock (1995), prix Ostrowski (1995), prix Fermat (1995), prix Wolf (1996), prix Cole (1997), prix du Clay Mathematics Institute (1999), prix Shaw (2005) et prix Abel en 2016.

2 Tests de primalité

Extrait de l'aide de Maple relative à "isprime(n)" :
It returns false if n is shown to be composite within one strong pseudo-primality test and one Lucas test and returns true otherwise. If isprime returns true, n is "very probably" prime - see Knuth "The art of computer programming", Vol 2, 2nd edition, Section 4.5.4, Algorithm P for a reference and H. Reisel, "Prime numbers and computer methods for factorization".
No counter example is known and it has been conjectured that such a counter example must be hundreds of digits long.

2.1 Test de Lucas-Lehmer (1891, 1927) ...

2.2 Test de primalité probabiliste de Miller-Rabin (1976-1980)

(Il s'agit d'un "test de forte pseudo-primalité").

Algorithme de Miller-Rabin : (étant donné un entier n, est-il premier ?)

Soit $n \in \mathbb{N}$, impair, et m un entier impair tel que $n-1 = 2^k m$.
Soit a un entier (aléatoire) tel que $1 < a < n$.

- Si $a^m \equiv 1 \mod n$ alors n est (peut-être) premier ;
- sinon, s'il existe $i \in [\![0 \ldots k-1]\!]$ tel que $a^{2^i m} \equiv -1 \equiv n-1 \mod n$ alors n est (peut-être) premier ;
- sinon n est (sûrement) factorisable.

Mise en œuvre pratique :

D'après le Théorème de raréfaction des nombres premiers, la probabilité d'erreur (réponse "n est premier" alors que n n'est pas premier) de l'algorithme de Miller-Rabin est inférieure (largement) à $\frac{1}{4}$.

Lorsque l'on "tire" e fois un nombre a, de façon aléatoire, $1 < a < n$, la probabilité pour que "n soit factorisable" , sachant que l'algorithme a répondu e fois "n est (peut-être) premier", est majorée par

$$\frac{\log n - 2}{\log n - 2 + 4^{e+1}}$$

On prévoit donc d'effectuer au plus e (par exemple $e = 30$ tirages ou plus si n est très très grand).

- Dès que l'algorithme répond "n est factorisable", on sait que n n'est pas premier.
- Si l'algorithme a répondu e fois "n est premier" alors la probabilité pour que ce soit faux est extrêmement faible.

Remarque. Bien souvent, on cumule ce test avec un autre test de (pseudo-)primalité.

Eléments d'étude de complexité :

- Le coût d'obtention de k et de m (fait une seule fois) est plus ou moins faible selon la structure de donnée utilisée pour représenter les grands entiers (quasi nul si les entiers sont représentés en binaire)
- Le calcul de $b = a^m \mod n$ a un coût en $O(\log_2 m)$ (exponentiation rapide)
- Le calcul de $a^{2^i m} = b^{2i} \mod n$ se fait avec un coût faible, puisque, à chaque itération on peut calculer $b^{2^i} = \left(b^{2^{i-1}}\right)^2 \mod n$, avec un coût de calcul égal à celui d'un carré et d'une division ($\mod n$) sur le résultat de l'itération précédente.

2.3 Test de primalité AKS (Agrawal-Kayal-Saxena, 2002) ...

Algorithme de preuve de primalité déterministe et généraliste (fonctionne pour tous les nombres) publié le 6 août 2002 par trois scientifiques indiens nommés Manindra Agrawal, Neeraj Kayal et Nitin Saxena. Premier test en mesure de déterminer la primalité d'un nombre dans un temps polynomial. Cette publication leur a valu le prestigieux prix Gödel 2006.

A développer ...

2.4 Autres ...

3 Factorisation (décomposition d'un entier en facteurs premiers)

On ne considère que le cas d'un entier N impair et on commence par rechercher un facteur d de N (pour une factorisation complète, il faudra recommencer pour d et N/d, qui sont plus "petits" que N).
Les petits nombres (au plus 12 décimales) se factorisent assez facilement ainsi que certains grands nombres quant ils ont des propriétés particulières ... mais, dans le cas général, la factorisation d'un grand nombre est un problème très difficile à réaliser en un temps fini.

3.1 Crible d'Eratosthène, 276 – 194 av. JC (pour mémoire)

On tente la division de l'entier N par les entiers impairs jusqu'à $\lfloor\sqrt{N}\rfloor$ ....
On peut améliorer l'algorithme en stockant les nombres premiers au fur et à mesure de leur découverte, ce qui donne une "base" (famille) de (petits) facteurs premiers, que l'on peut utiliser ultérieurement ou dans des algorithmes plus sophistiqués.
Cette méthode ne convient que pour des nombres N petits.

3.2 Méthode de factorisation de Fermat, 1640 (lente pour des grands nombres)

Soit N impair, se décomposant sous la forme $N = xy$ avec $1 < x \leqslant y$, (x et y inconnus, impairs).

$$\text{Si on pose } \begin{cases} u = \dfrac{x+y}{2} \\ v = \dfrac{y-x}{2} \end{cases}, \text{ on a } \begin{cases} x = u-v \\ y = u+v \end{cases} \quad \text{et } N = u^2 - v^2.$$

On va donc chercher u et v entiers tels que $N = u^2 - v^2$ et on retournera le couple $(u-v, u+v)$:

Algorithme :

- Initialisation : $u \leftarrow \lfloor\sqrt{N}\rfloor$, $v \leftarrow 0$, $r \leftarrow u^2 - N$ (c.a.d. $r \leftarrow u^2 - v^2 - N$)
- Tant que $r <> 0$ faire
  - si $r < 0$ alors $u \leftarrow u+1$, sinon $v \leftarrow v+1$
  - $r \leftarrow u^2 - v^2 - N$ (calcul un peu lourd)
- Retourner $(u-v, u+v)$

Algorithme amélioré : en utilisant $u' = 2u+1$ et $v' = 2v+1$,

- Initialisation : $u' \leftarrow 2\lfloor\sqrt{N}\rfloor + 1$, $v' \leftarrow 1$, $r \leftarrow \lfloor\sqrt{N}\rfloor^2 - N$
- Tant que $r <> 0$ faire
  - si $r < 0$ alors $r \leftarrow r + u'$ puis $u' \leftarrow u' + 2$
    sinon $r \leftarrow r - v'$ puis $v' \leftarrow v' + 2$
- Retourner $\left(\dfrac{u'-v'}{2}, \dfrac{u'+v'-2}{2}\right)$

Remarques.

- $y = u+v$ sera le plus petit facteur de N supérieur ou égal à $\sqrt{N}$
- $x = u-v$ sera le plus grand facteur de N inférieur ou égal à $\sqrt{N}$

et on pourra continuer à chercher la décomposition complète de N, en factorisant de même x et y ...

Preuves :

- Correction : Si l'algorithme se termine, il renvoie un résultat correct.
- Preuve de terminaison :
  On remarque que la variable de contrôle de boucle, r, ne varie pas de façon monotone,
  Il faut chercher autre chose ... variant de façon monotone, dans un domaine borné ...
- Complexité (de la deuxième version) : à chaque itération on effectue
  - un test de nullité et un test de signe (coûts négligeables devant l'addition)
  - deux additions d'entiers

  x étant le plus grand facteur de N inférieur ou égal à $\sqrt{N}$, on obtiendra x (et y) en $\lceil\sqrt{N}\rceil - x$ itérations, ce qui risque d'être très long si N est grand et n'admet qu'un "petit" facteur u.

En pratique, l'algorithme de Fermat n'est utilisable que pour des nombres pas trop grands, de la forme $N = xy$ avec x et y de même ordre de grandeur (c'est à dire proches de $\sqrt{N}$).
On aura intérêt à rechercher au préalable les "petits" facteurs premiers de N, dans une "base" de facteurs premiers connus, avant d'appliquer l'algorithme de Fermat au quotient.

3.3 Méthodes de Pollard

3.3.1 Méthode "p-1" de Pollard – 1974

Factorisation d'un entier admettant un facteur premier p (inconnu) tel que $p-1$ ne possède que de "petits" facteurs premiers.

Soit un entier n admettant un facteur premier p tel que $p-1$ ne possède que de petits facteurs premiers.
Soit B une borne telle que pour tout facteur premier u de $p-1$ on ait $u \leqslant B$.
Soit g un nombre compris entre 2 et $p-2$, (par exemple $g=2$ ou $g=3$).

Algorithme :

- $a \leftarrow g$
- Pour $k=2$ jusqu'à B, $a \leftarrow a^k \mod n$ (exponentielle rapide modulo n)
- Avec $d = pgcd(a-1,n)$, si $1 < d < n$ alors (d est un facteur de n) sinon (échec ...)

Remarque. En cas de succès, on peut continuer la décomposition de n en tentant la factorisation de n/d.

Preuve de la méthode "$p-1$" de Pollard ... voir Wikipédia.

Cette méthode permet (parfois) de trouver des facteurs de très grands nombres (mais les concepteurs de grands nombres se méfient ... et les choisissent tels qu'ils résistent à cet algorithme.

Remarque. Si on prend B trop grand, on perd de l'efficacité et, pour $B=\sqrt{n}$, l'algorithme se sera pas plus efficace que le crible d'Eratosthène.

La méthode "$p-1$" a été adaptée pour donner une méthode plus générale qui fait l'hypothèse beaucoup moins restrictive : "il existe un entier proche de p qui n'admet que des petits facteurs premiers" ....

3.3.2 Méthode "rho" de Pollard – 1975

Factorisation d'un entier admettant un "petit" facteur premier p (inconnu).

Soit $n=pq$ un entier, de facteur p inconnu.
Soit g une fonction pseudo-aléatoire simple à calculer (on prend généralement g telle que $g(x) = (x^2+1) \mod n$).
Soit $a \geqslant 1$ un entier (on prend généralement $a=1$).
Soient les suites (x_k) et $(y_k$, telles que $\begin{cases} x_0 = a & ; \quad x_{k+1} = g(x_k) \\ y_0 = g(x_0) & ; \quad y_{k+1} = g(g(y_k)) \quad (= x_{2k+2}) \end{cases}$

- On calcule $pgcd(|x_k - y_k|, n)$, pour $k \geqslant 0$, tant que ce pgcd est égal à 1.
- Si $pgcd(|x_k - y_k|, n) = n$ alors c'est un échec, sinon $pgcd(|x_k - y_k|, n)$ est un facteur de n.

Remarque. En cas d'échec, on peut recommencer avec une autre fonction g, une autre valeur de a ....

Preuve de la méthode "rho" de Pollard ... voir Wikipédia.

3.4 Autres méthode de recherche de facteurs premiers ...

- Factorisation par courbes elliptiques (Lenstra, 1985)
- Crible spécial de corps de nombres (SNFS, special number field sieve)

3.5 Exemples de "petits" nombres p premiers et de factorisation de $p-1$

$p = 124567171257792473713361$ $\quad p-1 = (2^4)(5)(11^2)(13)(37571083636607)(26347)$

$p = 238130989262374727$ $\quad p-1 = (2)(7)(23)(137)(245291)(1759)(12511)$

$p = 2060541199$ $\quad p-1 = (2)(3^2)(101)(227)(4993)$

$p = 2995882763$ $\quad p-1 = (2)(1497941381)$

$p = 2991487829$ $\quad p-1 = (2^2)(3)(7)(167)(163351)$

$p = 2742184769$ $\quad p-1 = (2^6)(277)(154681)$

$p = 2287449427$ $\quad p-1 = (2)(3)(131)(2910241)$

$p = 2169468083$ $\quad p-1 = (2)(101)(10739941)$

$p = 2452419443$ $\quad p-1 = (2)(11)(157)(710023)$

$p = 2889985793$ $\quad p-1 = (2^8)(11289007)$

$p = 2362913983$ $\quad p-1 = (2)(3^2)(11)(13)(917993)$

$p = 2284242749$ $\quad p-1 = (2^2)(571060687)$

$p = 2165493901$ $\quad p-1 = (2^2)(3)(5^2)(61)(73)(1621)$

$p = 2961523043$ $\quad p-1 = (2)(1997)(741493)$

$p = 2149126589$ $\quad p-1 = (2^2)(7)(457)(167953)$

$p = 2939029109$ $\quad p-1 = (2^2)(734757277)$

$p = 2590689529$ $\quad p-1 = (2^3)(3^3)(7)(59)(113)(257)$

$p = 486361033$ $\quad p-1 = (2^3)(3)(149)(277)(491)$

$p = 465702707$ $\quad p-1 = (2)(7)(29)(1147051)$

4 Logarithme discret

Le problème du logarithme discret est le suivant :

> Soit p un nombre premier et a un élément primitif de $\mathbb{Z}_p^*$ (a, générateur de $\mathbb{Z}_p^*$, est d'ordre $p-1$).
>
> Etant donné b élément de $\mathbb{Z}_p$, trouver l'unique x de $\mathbb{Z}_p$ tel que $a^x \equiv b \mod p$.
>
> (pour x de $\mathbb{Z}_p$ tel que $a^x \equiv b \mod p$, on note $x = \log_a b \mod p$).

Comme il s'agit d'un problème difficile à résoudre lorsque p est un entier très grand (au moins 150 chiffres), tel que $p-1$ ne possède que des "grands" facteurs premiers (cas où la méthode "$p-1$" de Pollar échoue), les logarithmes discrets sont très utilisée en cryptographie.

4.1 Algorithme de Pohlig-Hellman (1978)

4.1.1 Principe

Soit p un entier naturel premier, a un élément primitif de $\mathbb{Z}_p^*$.

> Si $p-1 = \prod_{i=1}^{k} q_i^{c_i}$ (décomposition de $p-1$ en facteurs premiers)
>
> - on calcule $x = \log_a b \mod q_i^{c_i}$ pour chacun des q_i,
> - et il ne reste plus qu'à appliquer le théorème des restes chinois pour avoir $x = \log_a b \mod (p-1)$
>
> (ce qui entraîne $x = \log_a b \mod p$)

4.1.2 Algorithme

1. Décomposition de $p-1$ en facteurs premiers ...
2. Calcul de $\log_a b \mod q^c$ lorsque q est un nombre premier (facteur d'ordre c de $p-1$).
   - Calculs préliminaires : Pour $i = 0 \cdots q-1$, $d_i \leftarrow a^{i(p-1)/q} \mod p$
   - Initialiser : $b_0 \leftarrow b$
   - Pour $j = 0 \cdots c-1$,
     - $d' \leftarrow b_j^{\frac{p-1}{q^{j+1}}} \mod p$
     - $u_j \leftarrow i$ où i est tel que $d' = d_i$
     - $b_j \leftarrow b_j \left(a^{-u_j q^j} \mod p \right) \mod p$
   - Résultat : $\log_a b \mod q^c = \sum_{i=0}^{c-1} u_i q^i$ (décomposition en base q)
3. Application du théorème des restes chinois ...

Preuve : Voir [St95]. Preuve utilisant la décomposition de $x \mod q^c$ en base q.

Remarque. Algorithme efficace lorsque $p-1$ ne possède que des petits facteurs premiers et qui peut devenir long, très long sinon :

- le calcul préliminaire construit un vecteur de taille q, ce qui risque d'être problèmatique lorsque l'entier q est trop "grand".
- on peut éviter la construction de ce vecteur, au prix du re-calcul systématique des d_i lors de la détermination de u_j (échange espace-mémoire contre temps de calcul).

4.2 Autres ...

5 Quelques références ...

[St95] D.R.STINSON. Cryptography Theory and Practice. CRC Press 1995
Traduction Française par S.Vaudenay. Cryptographie Théorie et Pratique
International Thomson Publishing France 1996.

[Sh ? ?] D. Shanks' undocumented square-free factorization.

[Po ? ?] J.M. Pollard's rho method.

[Le ? ?] Lenstra - Lenstra's elliptic curve method.

[Kn ? ?] Knuth ."The art of computer programming", Vol 2, 2nd edition.

[Re ? ?] H. Reisel, "Prime numbers and computer methods for factorization".

$< \mathscr{F}\mathscr{I}\mathscr{N} >$