École normale supérieure de Rennes

Université Paris Diderot

Stage de licence de mathématiques

Factorisation et primalité : une application de l'arithmétique

Auteur :
Adrian Petr

Encadrant :
M.Jean-François Mestre

Table des matières

1 Introduction

Dans sa célèbre œuvre de 1801 intitulée *Disquisitiones arithmeticae*, Carl Friedrich Gauss écrivait : "Le problème consistant à distinguer les nombres premiers des nombres composés, et à factoriser ces derniers en produit de facteurs premiers, est connu comme l'un des plus importants et des plus utiles en arithmétique" (où l'on considère qu'un nombre composé est un nombre supérieur à 2 et non premier). Ces dernières décennies, ce problème a requis d'autant plus d'attention que la sécurité des codes utilisés lors des échanges informatiques repose sur la grande difficulté à, voir l'impossibilité de, factoriser de très grands nombres. Dans ce travail, on présente et étudie les différentes méthodes qui permettent de déterminer si un entier donné est premier, et, s'il ne l'est pas, de déterminer sa décomposition en produit de facteurs premiers.

2 Chiffrement RSA

Le chiffrement RSA est un algorithme de cryptographie conçu en 1977 par Ronald Rivest, Adi Shamir et Leonard Adleman. Il est fréquemment utilisé dans le commerce électronique, et plus généralement dans l'échange de données confidentielles. Dans cette section, nous décrivons cet algorithme et montrons que sa sécurité repose sur la "difficulté" qu'ont les mathématiciens à factoriser un entier.

On commence par créer les couples de nombres qui seront utilisés pour coder et décoder les messages, en suivant le processus suivant.

1. Choisir p et q deux nombres premiers distincts
2. Calculer le produit :
$$n = pq$$
3. Calculer la valeur de l'indicatrice d'Euler ϕ en n :
$$\phi(n) = (p-1)(q-1)$$
4. Choisir un entier naturel e premier avec $\phi(n)$, et strictement inférieur à $\phi(n)$.
5. Calculer l'entier naturel d, inverse de e modulo $\phi(n)$, et strictement inférieur à $\phi(n)$

On dit que (n, e) est la clé publique du chiffrement, et que (n, d) est sa clé privée.

Avec la clé publique, on code un message, représenté par un entier naturel M premier avec n et strictement inférieur à n, en calculant le reste positif C de la

division de M^e par n. Pour déchiffrer le message codé C, il suffit pour le détenteur de la clé privée de calculer le reste positif de la division de C^d par n. En effet, par construction, il existe un entier k tel que :

$$ed = k\phi(n) + 1$$

Or on a par le théorème d'Euler :

$$M^{\phi(n)} \equiv 1 \mod n$$

Donc :

$$C^d \equiv M^{ed} \equiv M^{k\phi(n)+1} \equiv M \mod n$$

Remarque. *Les calculs de M^e et C^d peuvent être effectués "rapidement", c'est à dire avec une complexité polynomiale en la taille des entrées, grâce à l'algorithme d'exponentiation rapide.*

La sécurité de l'algorithme repose sur la confidentialité de la clé privée (n, d). L'entier n étant donné dans la clé publique, il s'agit de trouver d, inverse de e (qui est connu par la clé publique) modulo $\phi(n)$. Or, c'est justement le calcul de $\phi(n)$ qui peut s'avérer extrêmement "long", voir impossible pratiquement, lorsqu'on ne connaît pas les nombres premiers p et q qui composent n. En fait, la connaissance de $\phi(n)$ équivaut à celle des nombres p et q, car ces derniers sont les racines du polynôme $X^2 + (\phi(n) - n - 1)X + n$, dont les racines peuvent être calculées rapidement, par exemple avec la méthode de Newton. En définitive, la sécurité du chiffrement RSA repose donc sur la difficulté à factoriser un nombre en produit de facteurs premiers.

3 Algorithme AKS

Le test de primalité AKS est un algorithme publié en 2002 par Manindra Agrawal, Neeraj Kayal et Nitin Saxena ([Nai82]), qui permet de décider, de manière *déterministe et avec une complexité polynomiale en la taille de l'entrée*, si un nombre donné est premier ou non. Cet algorithme résout donc le problème *théorique* de la détermination de la primalité d'un nombre.

3.1 Principe

L'algorithme repose sur la généralisation suivante du petit théorème de Fermat, énonçant une condition nécessaire et suffisante pour qu'un nombre donné soit premier.

Théorème. *Soient a un entier et n un entier supérieur ou égal à 2 premier avec a. n est premier si et seulement si :*

$$(X+a)^n \equiv X^n + a \mod n$$

Preuve. Si i est entre 0 et n, le coefficient de X^i dans le polynôme $(X+a)^n - (X^n+a)$ est $\binom{n}{i}a^{n-i}$.

Supposons n premier. On a :

$$\forall 0 < i < n, \binom{n}{i} \equiv 0 \mod n$$

d'où l'égalité.

Supposons n composé. Soit q un facteur premier de n, et k la valuation q-adique de n. q^k ne divise pas $\binom{n}{q}$ et est premier avec a^{n-q}, de sorte que le coefficient de X^q dans le polynôme $(X+a)^n - (X^n+a)$ n'est pas congru à 0 modulo n. Ainsi on a :

$$(X+a)^n \not\equiv X^n + a \mod n$$

□

Dans la suite, si P, Q, R sont des polynômes de $\mathbb{Z}[X]$, et si n est un entier, on notera :

$$P \equiv Q \mod (R, n)$$

si les classes d'équivalences de P et Q dans $(\mathbb{Z}/n\mathbb{Z})[X]/(R)$ sont égales. L'algorithme se donne un entier $n > 1$ et procède comme suit :

1. S'il existe deux entiers naturels a et b strictement supérieurs à 1, tels que :

   $$n = a^b$$

   retourner "n est composé"
2. Trouver le plus petit entier r premier avec n tel que l'ordre de n dans le groupe multiplicatif des inversibles de $\mathbb{Z}/r\mathbb{Z}$, noté $o_r(n)$, soit strictement supérieur à $\log(n)^2$
3. S'il existe un entier a inférieur à r tel que :

   $$1 < a \wedge n < n$$

   retourner "n est composé"
4. Si n est inférieur à r, retourner "n est premier"

5. Pour a allant de 1 à $\lfloor\sqrt{\phi(r)}\log(n)\rfloor$, faire :

$$\text{si } (X+a)^n \not\equiv (X^n+a) \mod (X^r-1, n), \text{ retourner "}n \text{ est composé"}$$

6. Retourner "n est premier"

On va maintenant énoncer une série de lemmes qui permettront de montrer la correction de l'algorithme, autrement dit de montrer que l'algorithme décrit précédemment retourne "n est premier" si et seulement si n est effectivement premier.

Lemme. *Si n est premier, alors l'algorithme retourne "n est premier"*

Preuve. On suppose que n est premier. Il est alors impossible que les étapes 1 et 3 de l'algorithme retournent "n est composé". Par la généralisation du petit théorème de Fermat, il est par ailleurs impossible que la boucle "for" de l'étape 5 retourne "n est composé". Ainsi l'algorithme retournera nécessairement "n est premier".

□

Si l'algorithme retourne "n est premier" à l'étape 4, alors n est effectivement premier, car sinon l'étape 3 aurait trouvé un facteur non trivial de n et aurait retourné "n est composé". C'est pourquoi on se place dans la suite dans le dernier cas qu'il reste à traiter, autrement dit celui où l'algorithme retourne "n est premier" à l'étape 6.

On donne maintenant un lemme qui permet de borner l'entier r de l'étape 2.

Lemme. *Il existe un entier r tel que :*

$$r \leq max(3, \lceil\log(n)^5\rceil) \text{ et } o_r(n) > \log(n)^2$$

Preuve. Lorsque n est égal à 2, 3 satisfait trivialement les conditions du lemme. Supposons désormais n supérieur à 3. On a alors :

$$A = \lceil\log(n)^5\rceil > 10$$

Soit r le plus petit entier naturel qui ne divise pas la quantité :

$$N = n^{\lfloor\log(A)\rfloor} \prod_{i=1}^{\lfloor\log(n)^2\rfloor} (n^i - 1)$$

On a :

$$N < n^{\lfloor\log(A)\rfloor + \frac{1}{2}\log(n)^2(\log(n)^2-1)} \leq n^{\log(n)^4} \leq 2^{\log(n)^5} \leq 2^A$$

Comme A est supérieur à 7, le plus petit commun multiple des A premiers nombres est au moins 2^A ([Nai82]) ; or r est le plus petit des nombres qui ne divise pas N,

donc nécessairement r est inférieur à A. On note que $r \wedge n$ ne peut être divisible par tous les facteurs premiers de r, car sinon r diviserait $n^{\lfloor \log(A) \rfloor}$. En effet, chaque valuation p-adique k de r, pour p un facteur premier de r, vérifie :

$$p^k \leq A$$

car r est inférieur à A, et donc :

$$k \leq \lfloor \log(A) \rfloor$$

Ainsi, $\frac{r}{r \wedge n}$ ne divise pas non plus N. Par minimalité de r, on déduit que r et n sont premiers entre eux. De plus, r ne divise aucun des nombres de la forme $n^i - 1$, pour i entre 1 et $\lfloor \log(n)^2 \rfloor$, donc on a bien :

$$o_r(n) > \log(n)^2$$

□

Comme $o_r(n)$ est supérieur à 2, il existe un facteur premier p de n tel que $o_r(p)$ soit supérieur à 2. Nécessairement, p est strictement supérieur à r car sinon l'étape 3, ou l'étape 4, aurait déjà décidé de la primalité de n. Notons :

$$l = \lfloor \sqrt{\phi(r)} \log(n) \rfloor$$

Comme l'étape 5 ne retourne pas "n est composé", on a :

$$\forall 0 \leq a \leq l, (X + a)^n \equiv (X^n + a) \mod (X^r - 1, n)$$

Cela entraîne :

$$\forall 0 \leq a \leq l, (X + a)^n \equiv (X^n + a) \mod (X^r - 1, p)$$

D'après le petit théorème de Fermat généralisé, on a :

$$\forall 0 \leq a \leq l, (X + a)^p \equiv (X^p + a) \mod (X^r - 1, p)$$

Des deux dernières équations on déduit :

$$\forall 0 \leq a \leq l, (X + a)^{\frac{n}{p}} \equiv (X^{\frac{n}{p}} + a) \mod (X^r - 1, p)$$

On peut généraliser ce résultat en introduisant les ensembles :

$$\mathcal{I} = \{(\frac{n}{p})^i p^j \mid i, j \geq 0\} \text{ et } \mathcal{P} = \{\prod_{a=0}^{l} (X + a)^{e_a} \mid e_a \geq 0\}$$

On montre que pour tout nombre m de l'ensemble $\mathcal{I}$, et tout polynôme $f(X)$ de l'ensemble $\mathcal{P}$, on a :

$$f(X)^m \equiv f(X^m) \mod (X^r - 1, p)$$

On va maintenant définir deux groupes à partir des ensembles $\mathcal{I}$ et $\mathcal{P}$. Le premier groupe, noté G, est celui des classes d'équivalence des éléments de $\mathcal{I}$ modulo r. C'est un sous-groupe des inversibles de $\mathbb{Z}/r\mathbb{Z}$, dont le cardinal sera noté t. Comme G est engendré par n et p modulo r, et que $o_r(n)$ est strictement supérieur à $\log(n)^2$, on a :

$$t > \log(n)^2$$

Pour définir le second groupe, on a besoin d'un résultat sur les polynômes cyclotomiques sur $\mathbb{F}_p$.

Définition. *Soient p un nombre premier et r un entier naturel. On note $\mathbb{U}$ le groupe des racines primitives $r^{\text{èmes}}$ de l'unité dans le corps de décomposition de $X^r - 1$ sur $\mathbb{F}_p$. On appelle alors $r^{\text{ème}}$ polynôme cyclotomique sur $\mathbb{F}_p$ le polynôme :*

$$\Phi_{r,p}(X) = \prod_{\zeta \in \mathbb{U}} (X - \zeta)$$

Remarque. *Si p est un nombre premier et si r est un entier naturel, alors $\Phi_{r,p}(X)$ est un polynôme de $\mathbb{F}_p[X]$. En effet, en regroupant les racines $r^{\text{ème}}$ de l'unité (dans la clôture algébrique de $\mathbb{F}_p$) suivant leurs ordres, on obtient la formule :*

$$X^r - 1 = \prod_{d|r} \Phi_{d,p}(X)$$

qui permet de démontrer par récurrence sur r que chaque polynôme $\Phi_{r,p}(X)$ est à coefficients dans $\mathbb{F}_p$.

On énonce maintenant le résultat dont on a besoin.

Proposition. *Si p est un nombre premier et si r est un entier naturel, alors $\Phi_{r,p}(X)$ se décompose sur $\mathbb{F}_p[X]$ en produit de polynômes irréductibles de degré $o_r(p)$.*

Preuve. Soit h un facteur irréductible de $\Phi_{r,p}(X)$ dans $\mathbb{F}_p[X]$, et d don degré. L'extension $\mathbb{F}_p[X]/(h)$ de $\mathbb{F}_p$ est un corps de cardinal p^d. Or c'est aussi la plus petite extension de $\mathbb{F}_p$ qui contienne une racine primitive $r^{\text{ème}}$ de l'unité. Donc d est le plus petit entier tel que r divise $p^d - 1$, c'est à dire que d est l'ordre multiplicatif de p dans $\mathbb{Z}/r\mathbb{Z}$.

□

Soit $h(X)$ un des facteurs irréductibles de $\Phi_{r,p}(X)$ sur $\mathbb{F}_p[X]$. Le second groupe, noté $\mathcal{G}$, est celui des classes d'équivalences des éléments de $\mathcal{P}$ modulo $h(X)$ et p. Le lemme suivant permet de minorer le cardinal de $\mathcal{G}$.

Lemme.

$$|\mathcal{G}| \geq \binom{t+l}{t-1}$$

Preuve. On commence par remarquer que X est une racine primitive de l'unité dans le corps $\mathbb{F}_p[X]/(h)$, car $h(X)$ est un facteur de $\Phi_{r,p}(X)$.

On va maintenant montrer qu'à deux polynômes de $\mathcal{P}$ de degré strictement inférieur à t, il correspond exactement deux éléments de $\mathcal{G}$ par la surjection canonique. Soient $f(X)$ et $g(X)$ deux polynômes de $\mathcal{P}$ de degré strictement inférieur à t, égaux dans $\mathbb{F}_p[X]/(h)$. Soit m un élément de G. $f(X)^m$ et $g(X)^m$ sont toujours égaux dans $\mathbb{F}_p[X]/(h)$, et comme h divise $X^r - 1$, il en est de même pour $f(X^m)$ et $g(X^m)$. Ainsi X^m est une racine du polynôme :

$$Q(Y) = f(Y) - g(Y)$$

quelque soit l'élément m de G. Comme m et r sont premiers entre eux lorsque m est dans G, chaque X^m est une racine primitive $r^{\text{ème}}$ de l'unité dans $\mathbb{F}_p[X]/(h)$. Ainsi Q a t racines distinctes dans $\mathbb{F}_p[X]/(h)$. Or Q est de degré strictement inférieur à t par choix de f et g. Donc Q est nul, ce qui revient à dire que f et g sont égaux.

On remarque que les classes de i et j dans $\mathbb{F}_p$ sont différentes lorsque i et j sont deux entiers distincts entre 1 et l, car :

$$l = \lfloor \sqrt{\phi(r)} \log(n) \rfloor < \sqrt{r} \log(n) < r < p$$

Ainsi les éléments $X, X+1, ..., X+l$ sont tous distincts dans $\mathbb{F}_p[X]/(h)$. On a montré qu'il existe au moins $l+1$ polynômes distincts de degré 1 dans $\mathcal{G}$, donc il existe au moins $\binom{t+l}{t-1}$ polynômes distincts de degré strictement inférieur à t dans $\mathcal{G}$.

□

Lorsque n n'est pas une puissance de p, on peut aussi majorer la cardinal de $\mathcal{G}$.

Lemme. *Si n n'est pas une puissance de p, alors :*

$$|\mathcal{G}| \leq n^{\sqrt{t}}$$

Preuve. On suppose que n n'est pas une puissance de p et on considère le sous-ensemble suivant de $\mathcal{I}$:

$$\mathcal{J} = \{(\frac{n}{p})^i p^j \mid 0 \leq i, j \leq \lfloor t \rfloor\}$$

$\mathcal{J}$ a :

$$(\lfloor t \rfloor + 1)^2 > t$$

éléments distincts. Comme G est de cardinal t, au moins deux éléments de $\mathcal{J}$, notés m_1 et m_2, sont égaux modulo r. On a :

$$X^{m_1} \equiv X^{m_2} \mod (X^r - 1)$$

donc si $f(X)$ est un élément de $\mathcal{P}$, alors :

$$f(X)^{m_1} \equiv f(X^{m_1}) \equiv f(X^{m_2}) \equiv f(X)^{m_2} \mod (X^r - 1, p)$$

c'est à dire :

$$f(X)^{m_1} = f(X)^{m_2} \text{ dans } \mathbb{F}_p[X]/(h)$$

Ainsi, la classe de $f(X)$ dans $\mathcal{G}$ est une racine du polynôme :

$$Q'(Y) = Y^{m_1} - Y^{m_2}$$

dans $\mathbb{F}_p[X]/(h)$. $Q'(Y)$ a au moins $|\mathcal{G}|$ racines distinctes dans $\mathbb{F}_p[X]/(h)$, et est de degré :

$$max(m_1, m_2) \leq (\frac{n}{p} \times p)^{\lfloor t \rfloor} \leq n^t$$

d'où le résultat

□

On peut maintenant montrer le résultat suivant.

Lemme. *Si l'algorithme retourne "n est premier", alors n est effectivement premier.*

Preuve. D'après les lemmes précédents, on a :

$$|\mathcal{G}| \geq \binom{t+l}{t-1} \geq \binom{l+1+\lfloor \sqrt{t}\log(n) \rfloor}{\lfloor \sqrt{t}\log(n) \rfloor} \geq \binom{2\lfloor \sqrt{t}\log(n) \rfloor + 1}{\lfloor \sqrt{t}\log(n) \rfloor} \geq 2^{\lfloor \sqrt{t}\log(n) \rfloor} \geq n^{\sqrt{t}}$$

Comme $|\mathcal{G}|$ est inférieur à $n^{\sqrt{t}}$ si n n'est pas une puissance de p, n est une puissance de p. Il existe k inférieur à 1 tel que :

$$n = p^k$$

Si k était strictement supérieur à 1, l'algorithme aurait retourner "n est composé" à l'étape 1. Donc :

$$n = p$$

est premier.

□

Ceci termine la preuve de la correction de l'algorithme.

3.2 Complexité

Pour le calcul de la complexité de l'algorithme AKS, on suppose que l'addition, la multiplication et la division entre deux nombre de taille m peuvent être effectuées avec une complexité en $\mathcal{O}(m)$ ([VZGG99]), où :

$$\mathcal{O}(t(n)) = O(t(n) \times poly(n))$$

On suppose aussi que ces mêmes opérations entre deux polynômes de degré d avec des coefficients de taille m peuvent être effectuées avec une complexité en $\mathcal{O}(dm)$ ([VZGG99]). On peut alors montrer le théorème suivant.

Théorème. *L'algorithme AKS détermine si un entier donné n est premier avec une complexité en $\mathcal{O}(\log(n)^{\frac{21}{2}})$*

Preuve. On montre que la première étape de l'algorithme s'effectue avec une complexité en $\mathcal{O}(\log(n)^3)$ ([VZGG99]).

A la deuxième étape, on trouve un entier r tel que $o_r(n)$ soit strictement supérieur à $\log(n)^2$. Ceci peut être fait en testant pour des valeurs successives de r si :

$$\forall k \leq \log(n)^2, n^k \not\equiv 1 \mod r$$

Pour chaque r, ce test nécessite au plus $O(\log(n)^2)$ multiplications modulo r et donc ce fait avec une complexité en $\mathcal{O}(\log(n)^2 \log(r))$. Par le deuxième lemme de la section précédente, seulement $O(\log(n)^5)$ différentes valeurs de r sont nécessaires. Ainsi, la deuxième étape de l'algorithme se fait avec une complexité en $\mathcal{O}(\log(n)^7)$.

La troisième étape nécessite le calcul d'au plus r pgcd. Chacun de ces calculs s'effectuant par l'algorithme d'Euclide avec une complexité en $O(\log(n))$ ([VZGG99]), la complexité de l'étape 3 est en $O(\log(n)^6)$.

La complexité de la quatrième étape est tout simplement en $O(\log(n))$

Dans la cinquième étape, il s'agit de vérifier au plus $\lfloor \sqrt{\phi(r)} \log(n) \rfloor$ équations. Chaque équation requiert $O(\log(n))$ multiplications entre des polynômes de degré

r avec des coefficients de taille $O(\log(n))$. Ainsi, chaque équation peut être vérifiée en $\mathcal{O}(r\log(n)^2)$ étapes. La complexité de l'étape 5 est en :

$$\mathcal{O}(r\sqrt{\phi(r)}\log(n)^3 = \mathcal{O}(r^{\frac{3}{2}}\log(n)^3) = \mathcal{O}(\log(n)^{\frac{21}{2}})$$

Cette complexité domine celle des autres étapes et est de fait la complexité de l'algorithme.

□

4 Factorisation exponentielle

4.1 Méthode naïve

Soit n un entier composé. Pour déterminer le plus petit facteur premier p de n, il suffit de diviser n par chaque nombre supérieur à 2 et inférieur à n, jusqu'à trouver un résultat entier. Puisque p est inférieur à $\sqrt{n}$, on effectue dans le pire des cas environ $\sqrt{n}$ opérations.

On est donc loin d'une complexité polynomiale en la taille de l'entrée ($\log(n)$), et les calculs deviennent de fait rapidement impraticables lorsque n est grand. Il s'agit donc de penser à des méthodes plus efficaces pour factoriser.

4.2 Méthode ρ de Pollard

Dans cette section, on présente une méthode conçue par John M. Pollard en 1975 pour trouver le plus petit facteur premier q d'un nombre composé n, avec une complexité de l'ordre de $n^{\frac{1}{4}}$.

4.2.1 Principe et algorithme

L'algorithme repose sur l'idée suivante. On se donne un polynôme f à coefficients dans $\mathbb{Z}$ qui sert à modéliser le hasard, et un entier naturel x_0 strictement inférieur à n. On considère la suite $(x_k)_{k\in\mathbb{N}}$ définie de sorte que si k est un entier naturel, alors x_{k+1} est le reste positif de la division de $f(x_k)$ par n. On définit aussi la suite $(y_k)_{k\in\mathbb{N}}$ de sorte que :

$$\forall k \in \mathbb{N}, y_k = x_{2k}$$

$(y_k)_{k\in\mathbb{N}}$ vérifie la relation de récurrence :

$$\forall k \geq 0, y_{k+1} \equiv f \circ f(y_k) \mod n$$

On espère alors qu'il existera un entier t "pas trop grand" tel que :

$$x_t \neq y_t \text{ et } x_t \equiv y_t \bmod q$$

Ainsi on pourra expliciter q en calculant le plus grand commun diviseur de $(y_t - x_t)$ et n.

En pratique, l'algorithme se donne un polynôme f à coefficients dans $\mathbb{Z}$ et un entier x_0 strictement inférieur à n, et procède comme suit :

1. Poser y_0 le reste de la division euclidienne de $f(x_0)$ par n
2. Tant que $(y_k - x_k)$ et n sont premiers entre eux, ou que x_k et y_k sont égaux, calculer :
   x_{k+1}, le reste de la division euclidienne de $f(x_k)$ par n
   y_{k+1}, le reste de la division euclidienne de $f \circ f(y_k)$ par n
3. Retourner $(y_k - x_k) \wedge n$

4.2.2 Complexité

Dans cette partie, on présente le résultat appelé "paradoxe des anniversaires" qui permettra d'estimer le nombre d'itérations nécessaires pour que deux termes de la suite $x_0, ..., x_{N-1}$ soient égaux modulo q.

Théorème (Paradoxe des anniversaires généralisé)**.** *Soit E un ensemble fini non vide. La probabilité $p(N)$, que parmi N éléments de E tirés avec remise deux soient identiques, est donnée par la formule :*

$$p(N) = 1 - \frac{1}{|E|^N} \frac{|E|!}{(|E| - N)!}$$

où $|E|$ désigne le cardinal de E.

Preuve. Notons Ω l'ensemble E^N, et P la probabilité usuelle définie sur $\mathfrak{P}(\Omega)$. Le nombre $p(N)$ cherché est la probabilité de l'évènement :

$$A = \{(x_0, ..., x_{N-1}) \in \Omega \mid \exists 0 \leq i, j \leq n-1,\ i \neq j \text{ et } x_i = x_j\}$$

Or on a :

$$|\bar{A}| = |E| \times ... \times (|E| - N + 1) = \frac{|E|!}{(|E| - N)!}$$

où $\bar{A}$ désigne le complémentaire de A dans $\mathfrak{P}(\Omega)$. Donc on obtient bien :

$$p(N) = 1 - P(\bar{A}) = 1 - \frac{1}{|E|^N} \frac{|E|!}{(|E| - N)!}$$

□

On déduit de cette formule une valeur approchée lorsque $|E|$ est grand devant N :

$$p(N) \approx 1 - \exp(-\frac{N(N-1)}{2|E|})$$

Enfin on obtient une valeur approchée du nombre N d'éléments de E qu'il faut tirés pour que la probabilité que deux des N éléments soient égaux vaille p :

$$N(p) \approx \sqrt{2|E| \ln(\frac{1}{1-p})}$$

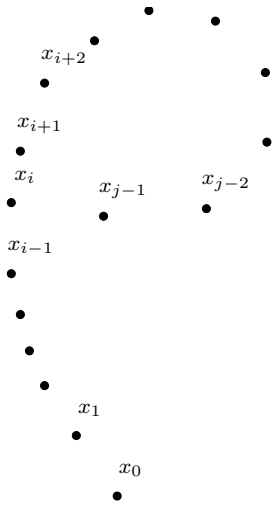
Maintenant, on utilise ce résultat pour déterminer le nombre N d'itérations nécessaires pour qu'il existe deux termes distincts de la suite $x_0, ..., x_{N-1}$ congrus modulo q, au moins avec une forte probabilité. E est l'ensemble $\mathbb{Z}/q\mathbb{Z}$ et les N éléments de E tirés avec remise sont les N classes d'équivalences de $x_0, ..., x_{N-1}$ modulo q ; donc la probabilité que deux termes parmi $x_0, ..., x_{N-1}$ soient égaux modulo q est supérieure à $\frac{1}{2}$ lorsque N est de l'ordre de $1.18\sqrt{q}$, et supérieure à 0.99 lorsque N est de l'ordre de $3\sqrt{q}$. Comme q est inférieur à $\sqrt{n}$, on a besoin d'environ $n^{\frac{1}{4}}$ itérations. On note j le plus petit entier naturel strictement positif vérifiant :

$$\exists 1 \leq i < j,\ x_i \equiv x_j \bmod q$$

Par l'estimation précédente, on sait que j est un $O(n^{\frac{1}{4}})$. Comme f est un polynôme à coefficients dans $\mathbb{Z}$, on a :

$$x_{i+1} \equiv x_{j+1} \bmod q$$

et ainsi de suite. Le nom de la méthode vient de la représentation graphique des termes de la suite : celle-ci dessine un ρ dont la queue contient les termes $x_0, ..., x_{i-1}$ et la boucle les termes $x_i, ...x_{j-1}$, de sorte qu'elle représente toutes les valeurs de la suite.



En notant :

$$l = j - i$$

la période de la boucle, on a :

$$\forall k \geq i,\ x_{k+l} \equiv x_k \bmod q$$

En choisissant un entier naturel m de sorte que ml soit inférieur à i, et en notant

$$t = ml$$

on a donc :

$$x_{2t} \equiv x_t \bmod q$$

avec t de l'ordre de i, c'est à dire en $O(n^{\frac{1}{4}})$. On voit qu'il suffit maintenant de calculer les plus grands communs diviseurs de $(x_k - x_{2k})$ et n pour k entre 1 et t, ce qui représente un nombre de calculs en $O(n^{\frac{1}{4}})$.

Finalement, on a bien montré que l'algorithme ρ de Pollard permet de trouver le plus petit facteur premier d'un nombre composé avec une complexité en $O(n^{\frac{1}{4}})$.

4.3 Comparaison pratique

Dans ce paragraphe, on présente le temps que mettent les deux méthodes ci-dessus à trouver un facteur non trivial d'un nombre n. On note $nextprime(m)$ le plus petit nombre premier supérieur à m, et on désigne par "#boucles" le nombre de boucles "while" que l'algorithme ρ exécute avant de trouver le nombre premier q qui divise le nombre à factoriser n.

Nombre à factoriser	Temps par la méthode naïve	Temps par la méthode ρ	$\frac{\#boucles}{\sqrt{q}}$
$nextprime(10^4)^2$	0ms	0ms	0.40
$nextprime(10^5)^2$	85ms	0ms	1.64
$nextprime(10^6)^2$	797ms	16ms	1.28
$nextprime(10^7)^2$	8s 206ms	16ms	2.03
$nextprime(10^8)^2$	1min 24sec	16ms	0.86
$nextprime(10^9)^2$	14min 40s	63ms	0.87
$nextprime(10^{10})^2$	2h 37min 8s	402ms	1.57
$nextprime(10^{11})^2$	>5h	1s 831ms	2.35
$nextprime(10^{12})^2$	>5h	2s 401ms	0.96
$nextprime(10^{13})^2$	>5h	5s 518ms	0.69
$nextprime(10^{14})^2$	>5h	17s 513ms	0.67
$nextprime(10^{15})^2$	>5h	33s 591ms	0.40
$nextprime(10^{16})^2$	>5h	22s 783ms	0.08
$nextprime(10^{17})^2$	>5h	11min 23s	0.78
$nextprime(10^{18})^2$	>5h	14min 37s	0.31

5 Crible quadratique

Dans cette partie, on présente une méthode qui permet de factoriser un entier n composé et impair avec une complexité sous-exponentielle, plus précisément avec un temps τ tel que, pour tous réels b et d strictement positifs, il existe a et c vérifiant :

$$a\log(n)^b \leq \tau(n) \leq cn^d \text{ lorsque } n \longrightarrow \infty$$

5.1 Principe

La méthode est motivée par la proposition suivante :

Proposition. *Un entier impair n est composé si et seulement s'il existe deux entiers u et v tels que :*

$$u^2 \equiv v^2 \mod n, u \not\equiv v \mod n \text{ et } u \not\equiv -v \mod n$$

Preuve. Supposons n composé. Soient a et b deux entiers impairs tels que :

$$n = ab \text{ et } 1 < a < b < n$$

Les entiers :

$$u = \frac{b+a}{2} \text{ et } v = \frac{b-a}{2}$$

vérifient alors les conditions de la proposition.

Supposons maintenant n premier. Soient u et v deux entiers tels que :

$$u^2 \equiv v^2 \mod n$$

Par lemme de Gauss, n divise $(u-v)$ ou $(u+v)$, donc on a bien :

$$u \equiv v \mod n \text{ ou } u \equiv -v \mod n$$

□

L'idée de l'algorithme est donc d'expliciter des nombres u et v tels que :

$$u^2 \equiv v^2 \mod n, u \not\equiv v \mod n \text{ et } u \not\equiv -v \mod n$$

Le calcul des entiers $(u-v) \wedge n$ et $(u+v) \wedge n$ fournira des diviseurs non triviaux de n. Pour cela, on recherche des entiers naturels x_i strictement inférieurs à n, tels que le produit des y_i soit un carré, où y_i désigne le reste positif de la division de x_i par n. Une fois déterminés de tels entiers, on pose :

$$u = \prod_i x_i \text{ et } v = \sqrt{\prod_i y_i}$$

On a alors :

$$u^2 \equiv v^2 \mod n$$

Remarque. *Si n n'est pas une puissance d'un nombre premier, alors il y a plus d'une chance sur 2 pour que u et v fournissent des diviseurs non triviaux de n, lorsque u et v sont premiers avec n. En effet, en notant :*

$$n = q_1^{f_1} ... q_d^{f_d}$$

la décomposition de n en produit de facteurs premiers, on a :

$$(u^2 \equiv v^2 \mod n) \Leftrightarrow (\forall 1 \leq i \leq d, u^2 \equiv v^2 \mod q_i^{f_i})$$

Par le théorème chinois, il y a 2^d possibilités pour u, et parmi elles seules deux (v et $-v$) ne fournissent pas de diviseurs non triviaux de n.

Dans l'algorithme, on retient des entiers x_i entre 0 et $n-1$ associés à des entiers y_i d'une certaine forme, forme qu'on définit maintenant.

Définition. *Soit m un entier naturel non nul. On dit qu'un entier est m-friable si tous ses diviseurs premiers sont parmi les m premiers nombres premiers.*

On retient donc les entiers x_i associés à des entiers y_i m-friables (où m est un entier naturel non nul fixé à l'avance). On dispose d'un résultat d'algèbre linéaire qui limite le nombre d'entiers y_i m-friables à chercher pour obtenir v.

Lemme. *Soit k un entier strictement supérieur à m. Si $y_1, ..., y_k$ sont des entiers naturels m-friables, alors il existe une sous famille non vide de $(y_i)_{1 \leq i \leq k}$ dont le produit des éléments est un carré.*

Preuve. Notons p_j le $j^{\text{ème}}$ nombre premier. Pour chaque entier i compris entre 1 et k, on écrit :

$$y_i = \prod_{j=1}^{m} p_j^{\alpha_{i,j}}$$

la décomposition de y_i en produit de facteurs premiers, et on note :

$$v(y_i) = (\alpha_{i,1}, ..., \alpha_{i,m})$$

Soit M la matrice à coefficients dans $\mathbb{F}_2$, de taille (k, m), dont l'élément de la $i^{\text{ème}}$ ligne et de la $j^{\text{ème}}$ colonne est la classe de $\alpha_{i,j}$ dans $\mathbb{F}_2$. Comme k est supérieur à $m+1$, les vecteurs ligne de M forment un système lié du $\mathbb{F}_2$-espace vectoriel $\mathbb{F}_2^m$. Il existe donc un k-uplet non nul $(\lambda_1, ..., \lambda_k)$ de $\mathbb{F}_2$ tel que :

$$\lambda_1 v(y_1) + ... + \lambda_k v(y_k) \equiv 0 \mod 2$$

En notant $i_1, ..., i_t$ les entiers i entre 1 et k tels que λ_i est non nul, le produit $y_{i_1}...y_{i_t}$ est un carré.

□

Ainsi, après avoir tiré un nombre k strictement supérieur à m d'entiers x_i associés à des entiers y_i m-friables, on dispose d'indices $i_1, ..., i_t$ entre 1 et k tels que $y_{i_1}...y_{i_t}$ soit un carré. Comme décrit précédemment, on note :

$$u = x_{i_1}...x_{i_t} \text{ et } v = \sqrt{y_{i_1}...y_{i_t}}$$

en espérant que $(u - v) \wedge n$ et $(u + v) \wedge n$ fournissent des diviseurs non triviaux de n.

L'algorithme procède donc en deux étapes :

1. Recherche de k entiers x_i associés à des entiers y_i m-friables, avec :

$$k > m$$

2. Recherche d'une sous-famille de (y_i) dont le produit des éléments est un carré.

5.2 Complexité

La complexité de l'algorithme dépend de l'entier m et du nombre M d'entiers x_i à tirer. En effet, la première étape de collecte des entiers m-friables consiste, pour chaque entier x_i tirer, à déterminer si y_i est m-friables ou non, ce qui correspond à une complexité en $O(m\log(n))$, d'où une complexité pour la première étape en $O(Mm\log(n))$. Quant à la deuxième étape, il s'agit comme on l'a vu de résoudre un système linéaire de taille m, ce qui correspond à une complexité en $O(m^3)$ si l'on procède selon la méthode du pivot de Gauss. Cette partie montre comment optimiser le choix de m et de M pour minimiser la complexité de l'algorithme. Pour cela on suit principalement le raisonnement trouvé dans [Knu98].

On introduit la probabilité $P(m,n)$ pour qu'un entier x choisi au hasard entre 0 et $n-1$ soit associé à un entier y m-friable. En tirant au hasard M entiers x entre 0 et $n-1$, on obtient en moyenne $MP(m,n)$ entiers y associés et m-friables. Il s'agit donc de choisir m et M de sorte que :

$$(Mm\log(n) \text{ soit minimal}) \text{ et } (MP(m,n) > m)$$

Pour cela, on aura besoin du lemme suivant.

Lemme. *On a l'implication suivante :*

$$(r = 2\lfloor\frac{\log(n)}{2log(p_m)}\rfloor) \Rightarrow (P(m,n) \geq \frac{m^r}{r!n})$$

Preuve. Il s'agit de minorer le cardinal de l'ensemble :

$$A = \{0 \leq x \leq n-1 \mid \exists (e_1,...,e_m) \in \mathbb{N}^m, y = p_1^{e_1}...p_m^{e_m}\}$$

On note :

$$n = q_1^{f_1}...q_d^{f_d}$$

la décomposition de n en produit de nombres premiers ; On va introduire les ensembles :

$$E = \{e = (e_1,...,e_m) \in \mathbb{N}^m \mid e_1 + ... + e_m \leq r \text{ et } \forall 1 \leq i \leq d, (\frac{p_1^{e_1}...p_m^{e_m}}{q_i}) = 1\}$$

et :

$$B_e = \{0 \leq x \leq n-1 \mid y = p_1^{e_1}...p_m^{e_m}\} \text{ pour } e \in E$$

et :

$$B = \bigcup_{e \in E} B_e$$

Il est clair que la réunion précédente est disjointe, et qu'on a :

$$B \subset A$$

Soient e un élément de E et x est un entier naturel inférieur à $n-1$. On a la relation :

$$(x \in B_e) \Leftrightarrow (x^2 \equiv p_1^{e_1}...p_m^{e_m} \mod n) \Leftrightarrow (\forall 1 \leq i \leq d, x^2 \equiv p_1^{e_1}...p_m^{e_m} \mod q_i^{f_i})$$

car :

$$(e_1 + ... + e_m \leq r) \Rightarrow (p_1^{e_1}...p_m^{e_m} < n)$$

Or, par hypothèse, $p_1^{e_1}...p_m^{e_m}$ est un carré modulo chacun des q_i, ce qui implique que $p_1^{e_1}...p_m^{e_m}$ est un carré modulo chacun des $q_i^{f_i}$, car :

$$a^{\frac{q^f-1}{2}} = (a^{\frac{q-1}{2}})^{(1+q+...+q^{f-1})}$$

En notant α_i les entiers tels que :

$$p_1^{e_1}...p_m^{e_m} \equiv \alpha_i^2 \mod q_i^{f_i}$$

on a alors :

$$(x \in B_e) \Leftrightarrow (\forall 1 \leq i \leq d, x^2 \equiv \alpha_i^2 \mod q_i^{f_i})$$

Avec le théorème chinois, cela montre que :

$$\forall e \in E, Card(B_e) = 2^d$$

On a donc :

$$Card(A) \geq 2^d Card(E)$$

et il s'agit maintenant de minorer le cardinal de E. On note :

$$E' = \{(e'_1, ..., e'_m, e''_1, ..., e''_m) \mid e'_1+...+e'_m \leq \frac{r}{2} \text{ et } e''_1+...+e''_m \leq \frac{r}{2} \text{ et } (\frac{p_1^{e'_1}...p_m^{e'_m}}{q_i}) = (\frac{p_1^{e''_1}...p_m^{e''_m}}{q_i})\}$$

et :

$$n_a = Card(\{(e'_1, ..., e'_m) \mid e'_1+...+e'_m \leq \frac{r}{2} \text{ et } p_1^{e'_1}...p_m^{e'_m} \equiv (-1)^{a_i} \mod q_i\} \text{ pour } a \in \{0,1\}^d$$

On montre que :

$$Card(E) \geq \frac{Card(E')}{\binom{r}{r/2}} = \frac{\sum_a n_a^2}{\binom{r}{r/2}}$$

Or :

$$\sum_a n_a = Card(\{(e'_1, ..., e'_m) \mid e'_1 + ... + e'_m \leq \frac{r}{2}\})$$

On peut monter par récurrence sur m que :

$$\forall s \in \mathbb{N}, Card(\{(k_1, ..., k_m) \mid k_1 + ... + k_m \leq s\}) = \binom{s+m}{s}$$

En effet, la formule est vrai lorsqu'on remplace m par 1, et si la formule est vrai pour un entier m quelconque, alors on a :

$$Card(\{(k_1, ..., k_{m+1}) \mid k_1 + ... + k_{m+1} \leq s\}) = \sum_{k_{m+1}=0}^{s} \binom{s - k_{m+1} + m}{s - k_{m+1}} = \sum_{k=0}^{s} \binom{k+m}{k}$$

Or on montre aisément (avec la formule de Pascal) par récurrence sur s la formule :

$$\sum_{k=0}^{s} \binom{k+m}{k} = \binom{s + (m+1)}{s}$$

On a donc :

$$\sum_a n_a = \binom{r/2 + m}{r/2}$$

puis :

$$\sum_a n_a^2 \geq \frac{\binom{r/2+m}{r/2}^2}{2^d} \geq \frac{m^r}{2^d (r/2)!^2}$$

ce qui permet d'aboutir au résultat souhaité.

□

Avec ce lemme, on voit maintenant qu'il suffit de choisir m et M de sorte que :

$$(Mm\log(n) \text{ soit minimal}) \text{ et } (\frac{Mm^r}{r!n} > m)$$

Comme m et M sont indépendants, il conviendra de choisir M de l'ordre de $r!nm^{1-r}$ une fois qu'on aura choisi m. On est donc maintenant ramené à choisir m en fonction de n de sorte que $r!m^{2-r}$ soit minimal. Or on a :

$$r \sim \frac{\log(n)}{\log(p_m)} \text{ lorsque } n \to \infty$$

et d'après le théorème des nombres premiers :

$$m = \pi(p_m) \sim \frac{p_m}{\log(p_m)} \text{ lorsque } n \to \infty$$

On a donc :

$$m \sim \frac{r n^{\frac{1}{r}}}{\log(n)} \text{ et } r \sim \frac{\log(n)}{\log(m \log(m))} \text{ lorsque } n \to \infty$$

de sorte qu'il est équivalent de choisir r et de choisir m. On cherche donc r qui minimise $r! \log(n)^r n^{\frac{2}{r}} r^{2-r}$. Comme :

$$r! \sim \sqrt{2\pi r}(\frac{r}{e})^r \text{ lorsque } n \to \infty$$

on cherche à minimiser $\exp(\frac{5}{2}\log(r) - r + r\ln\ln(n) + \frac{2}{r}\ln(n))$. Ceci revient à trouver r tel que :

$$\frac{5}{2r} - 1 + \ln\ln(n) - \frac{2}{r^2}\ln(n) = 0$$

c'est à dire tel que :

$$(\ln\ln(n) - 1)r^2 + \frac{5}{2}r - 2\ln(n) = 0$$

Le discriminant Δ de cette équation vérifie :

$$\Delta = \frac{25}{4} + 8\ln(n)(\ln\ln(n) - 1) \sim 8\ln(n)\ln\ln(n) \text{ lorsque } n \to \infty$$

ce qui montre qu'on doit choisir r de sorte que :

$$r \sim \frac{-\frac{5}{2} + \sqrt{\Delta}}{2(\ln\ln(n) - 1)} \sim \sqrt{\frac{2\ln(n)}{\ln\ln(n)}} \text{ lorsque } n \to \infty$$

Or, on a vu que la complexité $Mm\log(n)$ de la première étape vérifie :

$$Mm\log(n) \sim \frac{r!n}{m^{r-2}}\log(n) \sim \sqrt{2\pi}\exp(\frac{5}{2}\log(r) - r + \frac{2}{r}\ln(n) + (r-1)\ln\ln(n))$$

et on a :

$$\frac{5}{2}\log(r) - r + \frac{2}{r}\ln(n) + (r-1)\ln\ln(n) \sim \frac{2}{r}\ln(n) + r\ln\ln(n) \sim \sqrt{8}\ln(n)\ln\ln(n)$$

On a ainsi prouvé le théorème suivant :

Théorème. *Il existe un algorithme qui trouve un facteur non trivial d'un entier composé n avec une complexité probable en $O(n^{\epsilon(n)})$, où :*

$$\epsilon(n) = c\sqrt{\frac{\ln\ln(n)}{\ln(n)}}$$

avec c n'importe quelle constante supérieure à $\sqrt{8}$.

5.3 Collecte des entiers friables

Dans la pratique, on se limite à choisir des entiers x_i tels que :

$$0 \leq x_i^2 - n \leq n - 1$$

de sorte que :

$$y_i = x_i^2 - n$$

Alors, pour qu'un des entiers y_i soit divisible par un nombre premier p, il faut que n soit un carré modulo p, et que x_i soit une racine carrée de n modulo p. On va donc proposer des solutions aux problèmes suivants :

1. Déterminer si un entier n est un carré modulo un nombre premier p donné
2. Déterminer une racine carrée x d'un nombre n modulo un nombre premier p donné

Ainsi, l'algorithme constituera la liste P des nombres premiers inférieurs à une constante B tels que :

$$\forall p \in P, (\frac{n}{p}) = 1$$

et choisira les entiers x_i qui vérifient :

$$\forall p \in P, x_i^2 \equiv n \mod p$$

l'existence de tels entiers étant assuré par le théorème chinois.

5.3.1 Loi de réciprocité quadratique

Dans cette partie, on démontre la loi de réciprocité quadratique, qui relie les assertions "p un carré modulo q" et "q est un carré modulo p" lorsque p et q sont deux nombres premiers distincts. On se réfère principalement à [Ser70].

Symbole de Legendre

Définition. *Soit p un nombre premier impair. Si x un élément de $\mathbb{F}_p$, on appelle symbole de Legendre de x modulo p le nombre :*

$$(\frac{x}{p}) = x^{\frac{p-1}{2}}$$

Si a est un entier dont la classe dans $\mathbb{F}_p$ est x, on note :

$$(\frac{a}{p}) = (\frac{x}{p})$$

Si a est un entier, le symbole de Legendre de a ne prend que les valeurs -1,0 et 1. Plus précisément, il vaut -1 si a n'est pas un carré modulo p, 0 si a est divisible par p et 1 si a est un carré non nul modulo p.

Par ailleurs, on vérifie aisément la relation :

$$\forall x, y \in \mathbb{Z}, (\frac{xy}{p}) = (\frac{x}{p})(\frac{y}{p})$$

Montrons maintenant la proposition suivante.

Proposition. *Si p est un nombre premier impair, alors :*

$$(\frac{2}{p}) = (-1)^{\frac{p^2-1}{8}}$$

Preuve. Soit α est une racine primitive $8^{ème}$ de l'unité dans une clôture algébrique de $\mathbb{F}_p$. On a :

$$(\alpha + \alpha^{-1})^2 = \alpha^2 + \alpha^{-2} + 2\alpha\alpha^{-1} = \alpha^{-2}(\alpha^4 + 1) + 2 = 2 \text{ car } \alpha^4 = -1$$

Or si p est congru à 1, ou à -1, modulo 8, alors :

$$(\alpha + \alpha^{-1})^p = \alpha^p + \alpha^{-p} = \alpha + \alpha^{-1}$$

c'est à dire :

$$2^{\frac{p-1}{2}} = 1 \text{ dans } \mathbb{F}_p$$

Et si p est congru à 5, ou à -5, modulo 8, alors :

$$(\alpha + \alpha^{-1})^p = \alpha^5 + \alpha^{-5} = -(\alpha + \alpha^{-1})$$

c'est à dire :

$$2^{\frac{p-1}{2}} = -1 \text{ dans } \mathbb{F}_p$$

D'où le résultat.

□

Pour démontrer la loi, on commence par prouver deux lemmes sur des objets particuliers appelés *sommes de Gauss*.

Soit ζ une racine primitive $q^{ème}$ de l'unité dans une clôture algébrique de $\mathbb{F}_p$. Si x est un élément de $\mathbb{F}_q$, ζ^x a un sens car :

$$\zeta^q = 1$$

On appelle alors somme de Gauss le nombre :

$$G = \sum_{x \in \mathbb{F}_q} (\frac{x}{q})\zeta^x$$

On énonce maintenant le premier résultat sur cet objet.

Lemme.

$$G^2 = (\frac{-1}{q})q$$

Preuve. On a :

$$G^2 = \sum_{x,y\in\mathbb{F}_q} (\frac{xy}{q})\zeta^{x+y} = \sum_{x\in\mathbb{F}_q} \zeta^x \sum_{y\in\mathbb{F}_q} (\frac{y(x-y)}{q})$$

Or si y est non nul, alors :

$$(\frac{y(x-y)}{q}) = (\frac{-y^2}{q})(\frac{1-xy^{-1}}{q}) = (\frac{-1}{q})(\frac{1-xy^{-1}}{q})$$

Donc :

$$G^2 = (\frac{-1}{q}) \sum_{x\in\mathbb{F}_q} a_x\zeta^x \text{ où } a_x = \sum_{y\in\mathbb{F}_q^*} (\frac{1-xy^{-1}}{q})$$

On a :

$$a_0 = \sum_{y\in\mathbb{F}_q^*} (\frac{1}{q}) = q-1$$

Et si x est non nul, alors :

$$y \in \mathbb{F}_q^* \mapsto 1-xy^{-1}$$

est une bijection de $\mathbb{F}_q^*$ dans $\mathbb{F}_q\backslash\{1\}$, donc on a :

$$a_x = \sum_{y\in\mathbb{F}_q} (\frac{y}{q}) - (\frac{1}{q}) = -(\frac{1}{q}) = -1$$

car dans $\mathbb{F}_q^*$, il ya autant de carrés que de non-carrés. On a donc :

$$G^2 = (\frac{-1}{q})(q-1-\sum_{x\in\mathbb{F}_q^*} \zeta^x) = (\frac{-1}{q})q$$

□

On énonce maintenant le second lemme.

Lemme.

$$G^{p-1} = (\frac{p}{q})$$

Preuve. Comme on est en caractéristique p :

$$G^p = \sum_{x\in\mathbb{F}_q} (\frac{x}{q})\zeta^{xp} = \sum_{x\in\mathbb{F}_q} (\frac{xp^{-1}}{q})\zeta^x = (\frac{p^{-1}}{q})G = (\frac{p}{q})G$$

D'où le résultat.

□

Avec ces deux résultats, on peut démontrer le théorème de Gauss suivant.

Théorème (Loi de réciprocité quadratique)**.** *Si p et q sont deux nombres premiers impairs, alors :*

$$(\frac{p}{q}) = (-1)^{\frac{(p-1)(q-1)}{4}}(\frac{q}{p})$$

Preuve. D'après les lemmes précédents on a :

$$(\frac{p}{q}) = G^{p-1} = (G^2)^{\frac{p-1}{2}} = (\frac{-1}{q})^{\frac{p-1}{2}} q^{\frac{p-1}{2}} = (-1)^{\frac{(p-1)(q-1)}{4}}(\frac{q}{p})$$

D'où le théorème.

□

Symbole de Jacobi Le symbole de Jacobi est défini simplement par multiplicativité à partir du symbole de Legendre.

Définition. *Soit n un entier impair différent de 1, soit x un élément de $\mathbb{Z}/n\mathbb{Z}$. En notant :*

$$n = \prod_i p_i$$

la décomposition de n en produit de facteurs premiers (avec répétition des nombres premiers), on appelle symbole de Jacobi de x le nombre :

$$(\frac{x}{n}) = \prod_i (\frac{x}{p_i})$$

Comme pour le symbole de Legendre, si a est un entier et si x est sa classe d'équivalence modulo n, alors on note :

$$(\frac{a}{n}) = (\frac{x}{n})$$

A partir des résultats du paragraphe précédent, on montre la proposition et le théorème suivants.

Proposition. *Si n est un entier impair différent de 1, alors :*

$$(\frac{2}{n}) = (-1)^{\frac{n^2-1}{8}}$$

Preuve. On remarque que la fonction ω, qui à un entier impair n associe la classe de $\frac{n^2-1}{8}$ dans $\mathbb{Z}/2\mathbb{Z}$ vérifie :

$$\omega(nm) = \omega(n) + \omega(m)$$

En utilisant ce résultat et la proposition du paragraphe précédent, on peut écrire :

$$(\frac{2}{n}) = \prod_i (\frac{2}{p_i}) = \prod_i (-1)^{\omega(p_i)} = (-1)^{\omega(n)}$$

d'où le résultat.

□

Théorème. *Si a et b sont des entiers impairs différents de 1, alors :*

$$(\frac{a}{b}) = (-1)^{\frac{(a-1)(b-1)}{4}} (\frac{b}{a})$$

Preuve. On note :

$$a = \prod_i p_i \text{ et } b = \prod_j q_j$$

les décompositions de a et b en produit de facteurs premiers. On remarque que la fonction ϵ, qui à un entier impair n associe la classe de $\frac{n-1}{2}$ dans $\mathbb{Z}/2\mathbb{Z}$, vérifie :

$$\epsilon(nm) = \epsilon(n) + \epsilon(m)$$

En utilisant ce résultat et la loi de réciprocité quadratique, on a alors :

$$(\frac{a}{b}) = \prod_j (\frac{a}{q_j}) = \prod_{i,j} (\frac{p_i}{q_j}) = \prod_{i,j} (-1)^{\epsilon(p_i)\epsilon(q_i)} (\frac{q_j}{p_i}) = (-1)^{\epsilon(a)\epsilon(b)} (\frac{b}{a})$$

d'où le résultat.

□

Ce résultat va nous permettre de déterminer si un entier n est un carré modulo un nombre premier p impair. Pour cela, on va définir des suites $(j_k)_{k\in\mathbb{N}}$, $(r_k)_{k\in\mathbb{N}}$ et $(s_k)_{k\in\mathbb{N}}$. On pose :

$$j_0 = 1, r_0 = n\%p, s_0 = p$$

où $a\%b$ désigne l'unique entier naturel inférieur à $b-1$ et congru à a modulo b. Ensuite, on procède de la manière suivante. Si r_k est pair on pose :

$$j_{k+1} = (-1)^{\frac{s_k^2-1}{8}} j_k, r_{k+1} = \frac{r_k}{2}, s_{k+1} = s_k$$

et si r_k est impair on pose :

$$j_{k+1} = (-1)^{\frac{(r_k-1)(s_k-1)}{4}} j_k, r_{k+1} = s_k \% r_k, s_{k+1} = r_k$$

On remarque immédiatement que, si k est un entier naturel, alors :

1. $0 < r_k \Rightarrow r_{k+1} < r_k$
2. $r_k < s_k$
3. s_k est impair

On va maintenant montrer que si :

$$k_0 = inf(\{k \in \mathbb{N} \mid r_{k+1} = 0\})$$

alors r_{k_0} est égal à 1. Pour cela, on montre le résultat suivant :

Lemme. *Si l est un entier naturel inférieur à k_0, alors r_{k_0} divise r_{k_0-l} et s_{k_0-l}.*

Preuve. On procède par récurrence sur l. Le résultat est vrai lorsque l est nul, car r_{k_0+1} est nul et r_{k_0} est impair. On suppose maintenant le résultat vrai pour un entier naturel l inférieur à $k_0 - 1$. On a alors la dichotomie suivante. Si $r_{k_0-(l+1)}$ est pair, alors :

$$r_{k_0-(l+1)} = 2r_{k_0-l} \text{ et } s_{k_0-(l+1)} = s_{k_0-l}$$

Comme r_{k_0} divise r_{k_0-l} et s_{k_0-l} par hypothèse, r_{k_0} divise bien $r_{k_0-(l+1)}$ et $s_{k_0-(l+1)}$. Si à l'inverse $r_{k_0-(l+1)}$ est impair, alors il existe un entier naturel q tel que :

$$s_{k_0-(l+1)} = qr_{k_0-(l+1)} + r_{k_0-l} \text{ et } s_{k_0-l} = r_{k_0-(l+1)}$$

Comme r_{k_0} divise s_{k_0-l} par hypothèse, r_{k_0} divise $r_{k_0-(l+1)}$, ce qui permet de montrer que r_{k_0} divise aussi $s_{k_0-(l+1)}$. D'où le résultat.

□

Ainsi, on a montré que r_{k_0} divise r_0 et s_0. Or, il existe un entier naturel q tel que :

$$n = s_0 q + r_0 = pq + r_0$$

Donc r_{k_0} divise n et p, c'est à dire que r_{k_0} est égal à 1 car p est un nombre premier qui ne divise pas n.

On va maintenant montrer le lemme suivant.

Lemme. *Si k est un entier naturel alors :*

$$r_k > 0 \Rightarrow (\frac{n}{p}) = j_k(\frac{r_k}{s_k})$$

Preuve. On procède par récurrence sur k. Si k est nul le résultat est vrai. On suppose le résultat vrai à un rang k quelconque, et on suppose que r_{k+1} est non nul. Si r_k est pair, alors on a :

$$(\frac{n}{p}) = j_k(\frac{r_k}{s_k}) = j_k(\frac{2 \times (r_k/2)}{s_k}) = j_k(\frac{2}{s_k})(\frac{r_k/2}{s_k}) = j_{k+1}(\frac{r_{k+1}}{s_{k+1}})$$

Si r_k est impair, r_k est aussi différent de 1 car sinon r_{k+1} serait nul ; on a donc :

$$(\frac{n}{p}) = j_k(\frac{r_k}{s_k}) = j_k(-1)^{\frac{(r_k-1)(s_k-1)}{4}}(\frac{s_k}{r_k}) = j_{k+1}(\frac{r_{k+1}}{s_{k+1}})$$

D'où le résultat.

□

On a donc montré que :

$$(\frac{n}{p}) = j_{k_0}(\frac{r_{k_0}}{s_{k_0}}) = j_{k_0}(\frac{1}{s_{k_0}}) = j_{k_0}$$

Ainsi, l'algorithme qui calcule les suites $(j_k)_{k\in\mathbb{N}}$, $(r_k)_{k\in\mathbb{N}}$ et $(s_k)_{k\in\mathbb{N}}$ détermine si n est un carré modulo p avec une complexité de l'ordre de celle de l'algorithme d'Euclide étendu, c'est à dire en $O(\log(N)^2)$ si n et p sont inférieurs à N ([Rob]).

5.3.2 Extraction d'une racine carrée

On va ici décrire l'algorithme de Tonelli-Shanks, qui permet de trouver une solution en x à l'équation :

$$x^2 \equiv n \mod p$$

où p est un nombre premier impair, et n est un carré modulo p.

Notons (s, t) l'unique couple d'entiers tels que :

$$p - 1 = 2^s t \text{ avec } t \text{ impair}$$

$(\mathbb{Z}/p\mathbb{Z})^*$, muni de la multiplication, est un groupe cyclique engendré par un élément noté g. En notant U le groupe engendré par g^t, et V le groupe engendré par g^{2^s}, on peut voir $(\mathbb{Z}/p\mathbb{Z})^*$ comme le produit cartésien de U et V, par l'isomorphisme :

$$\phi : (u, v) \in U \times V \mapsto uv$$

Notons :

$$(u, v) = \phi^{-1}(n)$$

x est solution de l'équation :

$$x^2 \equiv n \mod p$$

si et seulement s'il existe a dans U et b dans V tels que :

$$(a, b) = \phi^{-1}(x), a^2 = u \text{ et } b^2 = v$$

On est donc ramener à déterminer des racines carrés de u et v. Dans V :

$$b = v^{\frac{t+1}{2}}$$

est une racine carré de v car t est impair. Reste à trouver une racine carré de u dans U. Soit h un générateur du groupe U : pour trouver un tel générateur, il suffit de trouver un entier m qui n'est pas un carré modulo p et de poser :

$$h = m^t$$

Soit k un entier naturel strictement inférieur à 2^s tel que :

$$u = h^k$$

Notons :

$$\alpha_0 + 2\alpha_1 + ... + 2^{s-1}\alpha_{s-1} = k$$

la décomposition de k en base 2. Comme u est un carré dans U, on sait que α_0 est nul. En déterminant le $(s-1)$-uplet $(\alpha_1, ..., \alpha_{s-1})$, l'élément :

$$a = h^{\alpha_1 + 2\alpha_2 + ... + 2^{s-2}\alpha_{s-1}}$$

sera une racine carré de u. Pour cela, on commence par calculer :

$$u^{2^{s-2}} = (h^{2^{s-1}})^{\alpha_1}$$

Comme h est d'ordre 2^s, il suffit de constater si cet élément est égal à 1 ou non pour savoir si α_1 est égal à 0 ou à 1. On recommence avec :

$$u' = uh^{-2\alpha_1} = h^{2^2\alpha_2 + ... + 2^{s-1}\alpha_{s-1}}$$

en calculant :

$$(u')^{2^{s-3}} = (h^{2^{s-1}})^{\alpha_2}$$

pour déterminer la valeur de α_2. On continue ainsi jusqu'à déterminer tous les nombres $\alpha_1, ..., \alpha_{s-1}$ qui permettent de calculer une racine carré de u.

L'algorithme procède donc ainsi :

1. Trouver les entiers s et t tels que :

$$p - 1 = 2^s t \text{ avec } t \text{ impair}$$

2. Trouver des entiers k et l tels que :

$$2^s k + tl = 1$$

avec l'algorithme d'Euclide étendu.

3. Trouver un générateur h de U
4. Trouver une racine carrée a de :

$$u = n^{tl}$$

5. Trouver une racine carrée b de

$$v = n^{2^s k}$$

6. Retourner ab, qui est une racine carrée de n modulo p

Cet algorithme permet donc de calculer une racine carrée d'un entier n modulo un nombre premier p avec une complexité probable en $O(\log(p)^3)$ (voir [Rob]).

5.4 Résolution du système linéaire

Dans cette partie, on suppose qu'on a collecté des entiers B-friables :

$$y_1 = \prod_{j=1}^{m} p_j^{\alpha_{1,j}}, ..., y_k = \prod_{j=1}^{m} p_j^{\alpha_{k,j}}$$

et qu'on dispose de la matrice :

$$M = (\alpha_{i,j} \mod 2) \in \mathcal{M}_{k,m}(\mathbb{F}_2)$$

Il s'agit maintenant de déterminer une sous-famille de $(y_i)_{1 \leq i \leq k}$ dont le produit des éléments est un carré, ce qui revient comme on l'a vu à résoudre le système linéaire :

$$M^t \Lambda = 0 \text{ où } \Lambda = (\lambda_1, ..., \lambda_k)^t \in \mathcal{M}_{k,1}(\mathbb{F}_2)$$

On note A la matrice de $\mathcal{M}_k(\mathbb{F}_2)$ dont les m premières lignes sont celles de M^t, et dont les $k - m$ dernières lignes sont remplies de 0. Il suffit alors de déterminer un élément du noyau de A car :

$$Ker(A) \subset Ker(M^t)$$

Pour cela, on va chercher à déterminer :

$$P = p_0 + p_1 X + ... + p_d X^d$$

le polynôme minimal de A. Comme A n'est pas inversible, p_0 est nul. Donc, si v est un élément aléatoire de $\mathcal{M}_{k,1}(\mathbb{F}_2)$, l'élément $p_1 v + p_2 Av + ... + p_d A^{d-1} v$ est soit nul, soit un élément du noyau de A. Pour trouver P, on va utiliser la suite $(a_n)_{n\in\mathbb{N}}$ définie par :

$$\forall n \in \mathbb{N}, a_n = A^n v$$

où v est un élément aléatoire de $\mathcal{M}_{k,1}(\mathbb{F}_2)$. Cette suite vérifie :

$$\forall j \in \mathbb{N}, \sum_{i=0}^{d} p_i a_{i+j} = 0$$

On va maintenant expliciter l'algorithme de Berlekamp-Massey, qui permet de trouver *rapidement* le polynôme minimal d'une suite récurrente linéaire. On commence par une définition.

Définition. *Soit $(a_n)_{n\in\mathbb{N}}$ une suite à valeurs dans un corps $\mathbb{K}$. On note I_a l'ensemble des polynômes P de $\mathbb{K}[X]$ tels que :*

$$\forall j \in \mathbb{N}, \sum_{i\geq 0} p_i a_{i+j} = 0$$

On dit que a est une suite récurrente linéaire si I_a n'est pas réduit à $\{0\}$

Remarque. *En notant δ l'endomorphisme de $\mathbb{K}^{\mathbb{N}}$ qui à une suite $(a_n)_{n\in\mathbb{N}}$ associe la suite $(a_{n+1})_{n\in\mathbb{N}}$, on a l'égalité :*

$$I_a = \{P \in \mathbb{K}[X] \mid P(\delta)(a) = 0\}$$

qui permet de montrer aisément que I_a est un idéal de $\mathbb{K}[X]$.

Pour trouver le polynôme minimal P_a d'une suite récurrente linéaire a, l'algorithme se donne en entrée un entier n supérieur au degré de P_a, et la liste $[a_0, ..., a_{2n-1}]$ des $2n$ premiers termes de la suite ; il procède alors comme suit.

1. Introduire les variables locales suivantes :

$$R_0 = X^{2n}$$

$$R_1 = \sum_{i=0}^{2n-1} a_{2n-1-i} X^i$$

$$V_0 = 0$$

$$V_1 = 1$$

2. Tant que n est inférieure au degré de R_1, faire :

   Noter (Q, R) le couple (quotient,reste) de la division euclidienne de R_0 par R_1.

   Noter V la quantité $V_0 - QV_1$

   Modifier les variables locales de la manière suivante :

$$V_0 := V_1, V_1 := V, R_0 := R_1, R_1 := R$$

3. Retourner le polynôme obtenu en divisant V_1 par son coefficient dominant.

On va maintenant montrer la correction de l'algorithme, c'est à dire que le polynôme retourné par l'algorithme de Berlekamp-Massey est bien P_a.

Dans la suite on notera :

$$R_0 = X^{2n} \text{ et } R_1 = \sum_{i=0}^{2n-1} a_{2n-1-i} X^i$$

On commence par montrer le lemme suivant.

Lemme. *Un polynôme V de $\mathbb{K}[X]$ de degré inférieur à n vérifie :*

$$\forall j \in \mathbb{N}, \sum_{i \geq 0} v_i a_{i+j} = 0$$

si et seulement s'il existe un polynôme R de degré strictement inférieur à n tel que :

$$VR_1 \equiv R \mod R_0$$

Preuve. En notant $(r_i)_{i \in \mathbb{N}}$ le polynôme R_1, on a :

$$V(X)R_1(X) = \sum_{i \geq 0} (\sum_{j=0}^{i} v_j r_{i-j}) X^i$$

soit :

$$V(X)R_1(X) = \sum_{i=0}^{2n-1} (\sum_{j=0}^{i} v_j a_{j+(2n-1-i)}) X^i + \sum_{i \geq 2n} (\sum_{j=0}^{i} v_j r_{i-j}) X^i$$

D'où :

$$V(X)R_1(X) \equiv R(X) \mod X^{2n} \text{ avec } R(X) = \sum_{i=0}^{2n-1} (\sum_{j=0}^{i} v_j a_{j+(2n-1-i)}) X^i$$

On voit alors que le degré de R est strictement inférieur à n si et seulement si :

$$\forall n \leq i \leq (2n-1), \sum_{j=0}^{i} v_j a_{j+(2n-1-i)} = 0$$

Comme V est de degré inférieur à n, ceci équivaut à :

$$\forall 0 \leq j \leq (n-1), \sum_{i \geq 0} v_i a_{i+j} = 0$$

Reste à prouver que cela équivaut encore à :

$$\forall j \in \mathbb{N}, \sum_{i \geq 0} v_i a_{i+j} = 0$$

Supposons que :

$$\forall 0 \leq j \leq (n-1), \sum_{i \geq 0} v_i a_{i+j} = 0$$

et montrons par récurrence sur j que :

$$\forall j \in \mathbb{N}, \sum_{i \geq 0} v_i a_{i+j} = 0$$

Le résultat est vrai pour les $n-1$ premiers rangs, ; supposons le maintenant vrai jusqu'à un rang j supérieur à $n-1$ et montrons le au rang $j+1$. On sait qu'il existe un polynôme P de $\mathbb{K}[X]$, unitaire et de degré inférieur à n tel que :

$$\forall j \in \mathbb{N}, \sum_{i \geq 0} p_i a_{i+j} = 0$$

donc :

$$\forall 0 \leq i \leq n, a_{i+j+1} = - \sum_{k=0}^{deg(P)-1} p_k a_{k+(i+j+1-deg(P))}$$

D'où :

$$\begin{aligned} \sum_{i \geq 0} v_i a_{i+j+1} &= - \sum_{i \geq 0} v_i \sum_{k=0}^{deg(P)-1} p_k a_{k+(i+j+1-deg(P))} \\ &= - \sum_{k=0}^{deg(P)-1} p_k \sum_{i \geq 0} v_i a_{i+(k+j+1-deg(P))} \end{aligned}$$

Or on a :

$$0 \leq k + j + 1 - deg(P) \leq j$$

car :

$$0 \leq k \leq deg(P) - 1 \leq n - 1 \leq j$$

Par hypothèse de récurrence, on a donc :

$$\sum_{i \geq 0} v_i a_{i+j+1} = 0$$

ce qui termine la preuve par récurrence de la première implication. L'implication réciproque étant immédiate, le résultat est démontré.

□

Trouver P_a est donc équivalent à trouver des polynômes U, V et R de $\mathbb{K}[X]$ tels que :

$$deg(R) < n, deg(V) \leq n \text{ et } UR_0 + VR_1 = R$$

avec V de degré minimum ; P_a sera alors l'unique polynôme de $\mathbb{K}[X]$ unitaire et proportionnel à V. Ces relations rappellent l'algorithme d'Euclide étendu appliqués aux polynômes R_0 et R_1. En effet, ce dernier réalise des équations du type :

$$U_k R_0 + V_k R_1 = R_k$$

en définissant les suites $(U_k)_{k\in\mathbb{N}}$, et $(V_k)_{k\in\mathbb{N}}$ par :

$$V_0 = 0, V_1 = 1, U_0 = 1, U_1 = 0$$

et :

$$V_{k+2} = V_k - Q_{k+1}V_{k+1}, U_{k+2} = U_k - Q_{k+1}U_{k+1}$$

où (Q_{k+1}, R_{k+2}) désigne le couple (quotient, reste) de la division euclidienne de R_k par R_{k+1}. L'algorithme de Berlekamp-Massey consiste en fait en l'algorithme d'Euclide étendu démarrant avec les polynômes R_0 et R_1, et restreint au calcul de la suite $(V_k)_{k\in\mathbb{N}}$. Or on a le lemme suivant.

Lemme. *(Kronecker, 1881)*

Soit k_0 un entier naturel. Si U, V et R sont des polynômes de $\mathbb{K}[X]$ tels que :

1. $UR_0 + VR_1 = R$
2. $deg(V) < deg(V_{k_0+1})$
3. $deg(R) < deg(R_{k_0-1})$

alors il existe un polynôme c de $\mathbb{K}[X]$ tel que :

$$U = cU_{k_0} \text{ et } V = cV_{k_0}$$

Preuve. La première chose à remarquer est la formule suivante, qui se démontre aisément par récurrence :

$$\forall k \geq 0, U_k V_{k+1} - U_{k+1}V_k = (-1)^k$$

Ainsi, il existe des polynômes c et d de $\mathbb{K}[X]$ tels que :

$$U = cU_{k_0} + dU_{k_0+1} \text{ et } V = cV_{k_0} + dV_{k_0+1}$$

D'où :

$$R = c(U_{k_0}R_0 + V_{k_0}R_1) + d(U_{k_0+1}R_0 + V_{k_0+1}R_1) = cR_{k_0} + dR_{k_0+1}$$

Supposons maintenant d non nul. Comme :

$$deg(V) < deg(V_{k_0+1})$$

on a :

$$deg(c) + deg(V_{k_0}) = deg(d) + deg(V_{k_0+1})$$

d'où :

$$deg(c) \geq deg(V_{k_0+1}) - deg(V_{k_0})$$

Or :

$$(deg(c)+deg(R_{k_0}))-(deg(d)+deg(R_{k_0+1}) = deg(V_{k_0+1})-deg(V_{k_0})+deg(R_{k_0})-deg(R_{k_0+1})$$

$$= deg(R_{k_0-1}) - deg(R_{k_0+1}) > 0$$

Donc :

$$deg(R) = deg(c) + deg(R_{k_0}) \geq deg(V_{k_0+1}) - deg(V_{k_0}) + deg(R_{k_0})$$

Il s'agit maintenant de remarquer que la quantité $deg(R_k) + deg(V_{k+1})$ est indépendante de k car :

$$deg(R_{k+1}) = deg(R_k) - deg(Q_{k+1}) \text{ et } deg(V_{k+2}) = deg(V_{k+1}) + deg(Q_{k+1})$$

De ce fait on a :

$$deg(R) \geq deg(R_{k_0-1})$$

ce qui est contraire à l'hypothèse. Donc d est nul et on obtient le résultat.

□

Avec ce lemme, on est maintenant en mesure de montrer la correction de l'algorithme de Berlekamp-Massey. On note :

$$k_0 = inf(\{k \in \mathbb{N} \mid deg(R_k) < n\})$$

On a vu dans la preuve du lemme précédent que la quantité $deg(R_k) + deg(V_{k+1})$ est constante égale à :

$$deg(R_0) + deg(V_1) = 2n$$

Ainsi, le degré de V_{k_0+1} est strictement supérieur à n. En résumé, il existe des polynômes U et R de $\mathbb{K}[X]$ tels que :

1. $UR_0 + P_aR_1 = R$
2. $deg(P_a) \leq n < deg(V_{k_0+1})$
3. $deg(R) < n \leq deg(R_{k_0-1})$

D'après le lemme précédent, il existe donc un polynôme c tel que :

$$U = cU_{k_0} \text{ et } P_a = cV_{k_0}$$

Or P_a et R sont premiers entre eux, donc c est une constante. Ceci termine la preuve de la correction de l'algorithme de Berlekamp-Massey.

Remarque. *Dans le cas auquel on s'intéresse, où l'on cherche à trouver le noyau d'une matrice "creuse" A de $\mathcal{M}_k(\mathbb{F}_2)$, l'algorithme de Berlekamp-Massey permet de conclure avec une complexité en $O(\gamma k^2)$, où γ désigne le nombre de zéros dans M (voir [Tho03]).*

6 Annexes

6.1 Algorithme AKS

On donne ici les lignes de code, à exécuter sous Pari, de l'algorithme AKS décrit dans la première section.

```
log2(n)=
{
return(log(n)/log(2))
}

etape1(n)=
{
B=False;
b=2;
while(b<floor(log2(n))+1 && B==False,
  c=log2(n)/b;
  a=2^c;
  if(a==floor(a),B=True);
    b=b+1);
return(B)
}

ordre(r,n)=
{
k=0;
if(gcd(r,n)==1,
k=1;
```

```
while((n^k)%r<>1,
k=k+1));
return(k)
}

etape2(n)=
{
r=2;
while(r<max(3,(floor(log2(n))+1)^5) && ordre(r,n)<log2(n)^2+1,
r=r+1);
return(r)
}

etape3(r,n)=
{
B=False;
a=1;
while(a<r+1 && B==False,
  if(gcd(a,n)>1 && gcd(a,n)<n,
    B=True);
  a=a+1);
return(B)
}

etape4(r,n)=
{
B=False;
if(n<r+1,
  B=True);
return(B)
}

congrus(f,g,n)=
{
b=True;
k=0;
d=max(poldegree(f),poldegree(g));
while(k<d+1 && b==True,
if(polcoeff(f,k)%n<>polcoeff(g,k)%n,
b=False);
k=k+1);
return(b)
}
```

```
66
67 etape5(r,n)=
68 {
69 B=False;
70 a=1;
71 while(a<floor(sqrt(eulerphi(r))*log2(n))+1 && B==False,
72 f=((x+a)^n)%(x^r-1);
73 g=(x^n+a)%(x^r-1);
74 if(congrus(f,g,n)==False,
75 B=True);
76 a=a+1);
77 return(B)
78 }
79
80 aks(n)=
81 {
82 if(etape1(n)==True,
83   return(compose));
84 r=etape2(n);
85 if(etape3(r,n)==True,
86   return(compose));
87 if(etape4(r,n)==True,
88   return(premier));
89 if(etape5(r,n)==True,
90   return(compose));
91 return(premier)
92 }
```

6.2 Algorithme ρ de Pollard

On donne ici les lignes de code, à exécuter sous Pari, de l'algorithme ρ de Pollard décrit dans la deuxième section.

```
1 f(x)=
2 {
3 return(x^2+1)
4 }
5
6 pollard(n,u)=
7 {
8 x=u;
9 y=f(x)%n;
```

```
while(gcd(y-x,n)==1 || x==y,
  x=f(x)%n;
  y=f(f(y))%n);
return(gcd(y-x,n))
}
```

6.3 Algorithmes du crible quadratique

On donne ici les lignes de code, à exécuter sous Pari, des algorithmes décrit dans la troisième section.

6.3.1 Symbole de Legendre

L'algorithme ci-dessous calcule le symbole de Legendre d'un entier n modulo un nombre premier p impair.

```
symbole_legendre(n,p)=
{
if(n%p==0,
  return(0));
j=1;
r=n%p;
s=p;
while(r>1,
  t=r%2;
  if(t==0,
    j=(-1)^((s^2-1)/8)*j;
    r=r/2);
  if(t==1,
    j=(-1)^((r-1)*(s-1)/4)*j;
    u=r;
    r=s%r;
    s=u));
return(j)
}
```

6.3.2 Algorithme de Tonelli-Shanks

On donne ici les lignes de code de l'algorithme de Tonelli-Shanks.

```
etape1(p)=
{
s=0;
t=p;
while(t%2==0,
  t=t/2;
  s=s+1);
return([s,t])
}

etape2(c,d)=
{
r_0=c;
r_1=d;
k_0=1;
k_1=0;
l_0=0;
l_1=1;
while(r_1>1,
  r=r_0%r_1;
  q=(r_0-r)/r_1;
  k=k_0-q*k_1;
  k_0=k_1;
  k_1=k;
  l=l_0-q*l_1;
  l_0=l_1;
  l_1=l;
  r_0=r_1;
  r_1=r);
return([k_1,l_1])
}

etape3(p,t)=
{
m=0;
B=False;
while(B==False,
  r=random(p);
  if(symbole_legendre(r,p)==-1,
    B=True;
    m=r));
return(m^t%p)
}
```

```
44
45 etape4(u,s,p,h)=
46 {
47 u_0=u;
48 a=1;
49 i=2;
50 while(i<s+1,
51   if(u_0^(2^(s-i))%p<>1,
52     a=a*h^(i-1)%p;
53     u_0=u_0*h^(2^(s+1)-2)%p);
54   i=i+1);
55 return(a%p)
56 }
57
58 etape5(v,t,p)=
59 {
60 return(v^((t+1)/2)%p)
61 }
62
63 tonelli(n,p)=
64 {
65 [s,t]=etape1(p-1);
66 [k,l]=etape2(2^s,t);
67 h=etape3(p,t);
68 [u,v]=[n^(t*l)%p,n^(2^s*k)%p];
69 a=etape4(u,s,p,h);
70 b=etape5(v,t,p);
71 return(a*b%p)
72 }
```

6.3.3 Algorithme de Berlekamp-Massey

On donne ici les lignes de code de l'algorithme de Berlekamp-Massey ; a désigne la liste $[a_0, a_1, ..., a_{2n-1}]$ des $2n$ premiers termes de la suite $(a_k)_{k\in\mathbb{N}}$.

```
1 berlekamp(n,a)=
2 {
3 R_0=x^{2*n};
4 R_1=Pol(a,x);
5 V_0=0;
6 V_1=1;
7 while(poldegree(R_1)<n+1,
8   R=R_0%R_1;
```

```
  Q=(R_0-R)/R_1;
  V=V_0-Q*V_1;
  V_0=V_1;
  V_1=V;
  R_0=R_1;
  R_1=R);
v_1=polcoeff(V_1,poldegree(V_1));
return(V_1/v_1)
}
```

Références

[Knu98] Donald Ervin Knuth. *The Art of Computer Programming*. Addison Wesley Longman, 1998.

[Nai82] M. Nair. On chebyshev-type inequalities for primes. *Amer. Math. Monthly*, 1982.

[Rob] X.-F. Roblot. Arithmétique algorithmique.

[Ser70] Jean-Pierre Serre. *Cours d'Arithmétique*. Presses Universitaires de France, 1970.

[Tho03] Emmanuel Thomé. Berlekamp-Massey matriciel rapide, résolution de gros systèmes linéaires par "block Wiedemann", 2003.

[VZGG99] Joachim Von Zur Gathen and Jürgen Gerhard. *Modern Computer Algebra*. Cambridge University Press, 1999.