

Factorisation des entiers

Vincent Dalsheimer Gaëtan Pradel

Année 2016 - Semestre 10

Table des matières

1	Introduction	2
2	Un algorithme de chiffrement utilisant une clef publique $n = pq$: RSA	2
2.1	Création des clefs	3
2.2	Chiffrement	3
2.3	Déchiffrement	3
2.4	Justification	3
3	Méthode de factorisation naïve	4
4	L'algorithme $p - 1$ de Pollard	5
4.1	Principe de l'algorithme	5
4.2	Pseudo-code	6
4.3	Exemple	6
4.4	Les limites	7
5	Crible de Dixon	8
5.1	Principe de l'algorithme	8
5.2	Exemple	9
5.3	Pseudo-code	9
5.4	Précisions sur l'algorithme	9
5.4.1	Facteur non trivial	9
5.4.2	Choix de B	11
5.5	Limites de l'algorithme	13
6	Crible quadratique	13
6.1	Principe de l'algorithme	13
6.2	Pseudo-code	14
6.3	Exemple	14

1 Introduction

Factoriser un entier de manière efficace est un vieux problème d'arithmétique. Nombre de célèbres mathématiciens s'y sont confrontés (Ératosthène, Fermat, Gauss, Mersenne, ...) mais leur intérêt était purement mathématique.

Toutefois, avec l'apparition de la cryptologie, le problème prend une toute autre ampleur. En effet, certains algorithmes de chiffrement reposent sur une clef privée qui est la factorisation de la clef publique n (RSA par exemple). Réussir à factoriser n implique donc de pouvoir casser ces algorithmes.

Théorème 1 (Théorème fondamental de l'arithmétique). *Tout entier strictement positif peut être écrit comme un produit de nombres premiers d'une unique façon, à l'ordre près des facteurs.*

Ce théorème implique donc que tout entier n peut être factorisé de manière unique, justifiant alors la recherche d'algorithmes de factorisation. La performance de ces algorithmes est cruciale en cryptologie. En effet, c'est grâce à celle-ci que l'on peut déterminer la taille minimum nécessaire des clefs des algorithmes de chiffrement afin qu'ils ne soient pas cassables en un temps raisonnable.

Le plus ancien de ces algorithmes est la méthode de factorisation dite naïve. C'est la méthode de factorisation la plus intuitive. Depuis, par souci d'efficacité, de nombreux algorithmes plus rapides ont été trouvés, comme l'algorithme $p - 1$ de Pollard, le crible de Dixon ou encore le crible quadratique.

Nous étudierons par la suite le chiffrement RSA, ainsi que les quatre algorithmes sus-mentionnés.

2 Un algorithme de chiffrement utilisant une clef publique $n = pq$: RSA

Le chiffrement RSA est un chiffrement asymétrique, c'est-à-dire qu'il utilise une paire de clefs, qui sont des nombres entiers, composée d'une clef publique pour chiffrer et d'une clef privée pour déchiffrer. La clef privée peut être aussi utilisée pour signer un message. La clef publique est de la forme $n = pq$, et la clef privée peut être trouvée à partir de p et q . Ainsi, factoriser n revient à casser RSA.

2.1 Création des clefs

- Choisir p et q , deux nombres premiers entiers distincts ;
- calculer leur produit $n = pq$
- calculer $\phi(n) = (p - 1)(q - 1)$ qui est la valeur de l'indicatrice d'Euler en n ;
- choisir un entier naturel e premier avec $\phi(n)$ et strictement inférieur à $\phi(n)$;
- calculer l'entier naturel d , inverse de e modulo $\phi(n)$, et strictement inférieur à $\phi(n)$; d peut se calculer efficacement par l'algorithme d'Euclide étendu.

Le couple (n, e) est la clef publique et d est la clef privée.

La sécurité de cet algorithme repose sur le fait que la factorisation (de grands nombres) est un problème difficile, c'est-à-dire qu'on ne peut pas le résoudre en temps polynomial.

2.2 Chiffrement

Soit m un message à chiffrer. Le message chiffré c de m sera :

$$c \equiv m^e \pmod{n}.$$

2.3 Déchiffrement

Soit le message chiffré c comme ci-dessus. Pour retrouver le message m on fait :

$$c^d \equiv m^{ed} \equiv m \pmod{n}.$$

2.4 Justification

La démonstration repose sur le petit théorème de Fermat, à savoir que comme p et q sont deux nombres premiers, si m n'est pas un multiple de p on a la première égalité ci-dessous, et la seconde s'il n'est pas un multiple de q :

$$m^{p-1} \equiv 1 \pmod{p}, \quad m^{q-1} \equiv 1 \pmod{q}.$$

En effet

$$c^d \equiv (m^e)^d \equiv m^{ed} \pmod{n}.$$

Or

$$ed \equiv 1 \pmod{(p-1)(q-1)}$$

ce qui signifie que pour un entier k

$$ed = 1 + k(p-1)(q-1),$$

donc, si m n'est pas multiple de p d'après le petit théorème de Fermat

$$m^{ed} \equiv m^{1+k(p-1)(q-1)} \equiv m \cdot (m^{p-1})^{k(q-1)} \equiv m \pmod{p}$$

et de même, si m n'est pas multiple de q

$$m^{ed} \equiv m \pmod{q}.$$

3 Méthode de factorisation naïve

Pour factoriser un entier n , la méthode la plus naïve et la plus naturelle consiste à faire les divisions euclidiennes successives de n par les entiers i , $2 \leq i \leq \lfloor \sqrt{n} \rfloor$. Si le reste est nul, on garde i de côté et on recommence avec $\frac{n}{i}$. Sinon, on passe à $i + 1$. Quand $\frac{n}{i} = 1$, la factorisation est terminée et les facteurs de n sont donc les i qu'on a mis de côté.

Il est toutefois possible d'améliorer légèrement cet algorithme à l'aide du crible d'Ératosthène. Ce crible permet d'obtenir tous les nombres premiers inférieurs à un entier n . On prend la liste de tous les entiers compris entre 2 et n , puis on garde 2 mais on supprime tous ses multiples. On recommence la même opération avec l'entier le plus proche à ne pas avoir été déjà supprimé. On continue ainsi jusqu'à $\lfloor \sqrt{n} \rfloor$, entier à partir duquel tous les nombres restants dans la liste sont des premiers.

Une fois la liste des nombres premiers inférieurs à n générée, on applique la même méthode de factorisation, mais en divisant n non plus par les entiers i , $2 \leq i \leq \lfloor \sqrt{n} \rfloor$, mais par les premiers obtenus grâce au crible d'Ératosthène.

Pour des petits facteurs (moins de dix chiffres), la méthode naïve reste la méthode de factorisation la plus efficace. Pour des facteurs plus grands, cet algorithme est généralement moins efficace que ceux que nous allons étudier en suivant.

4 L'algorithme $p - 1$ de Pollard

L'algorithme $p - 1$ de Pollard est un algorithme de décomposition en produit de facteurs premiers. Cette méthode fonctionne seulement avec des nombres qui ont une forme particulière. Il trouve les facteurs p dont $p - 1$ est ultrafriable.

Définition 1 (Entier friable). *Un entier strictement positif n est dit B -friable ou B -lisse si tous ses facteurs premiers sont inférieurs ou égaux à B .*

Exemple 1. $90 = 2 \times 3^2 \times 5$ est 5-friable car aucun de ses facteurs premiers ne dépasse 5. Cette définition inclut les nombres qui ne figurent pas parmi les facteurs premier : par exemple, 12 est 5-friable.

Définition 2 (Entier ultrafriable). *Un nombre n est dit B -superlisse ou B -ultrafriable si toute puissance p^r d'un nombre premier qui divise n vérifie :*

$$p^r \leq B.$$

Exemple 2. $720 = 2^4 \times 3^2 \times 5$ est 5-friable mais pas 5-ultrafriable ($3^2 = 9 > 5$). Par contre il est 16-ultrafriable puisque sa plus grande puissance de facteur premier est $2^4 = 16$.

Lemme 1. *Si m est B -ultrafriable pour un certain seuil B , alors $m \mid \text{pgcd}(1, \dots, B)$.*

Lemme 2 (Lemme de Gauss). *Si un nombre entier a divise le produit de deux autres nombres entiers b et c , et si a est premier avec b , alors a divise c .*

4.1 Principe de l'algorithme

Soit n un entier divisible par un nombre premier p , avec $n \neq p$.

Théorème 2 (Petit Théorème de Fermat). *Si p est un nombre premier et si a est un entier non divisible par p , alors $a^{p-1} - 1$ est un multiple de p . C'est-à-dire :*

$$a^{p-1} \equiv 1 \pmod{p}.$$

Par le petit théorème de Fermat, nous savons que

$$a^{p-1} \equiv 1 \pmod{p}$$

pour a premier avec p .

Cela implique que pour tout multiple M de $p-1$ on a :

$$a^M - 1 \equiv 0 \pmod{p} \text{ car } a^{k(p-1)} - 1 = (a^{p-1} - 1) \sum_{i=0}^{k-1} a^{i(p-1)}.$$

D'après le lemme 1 si l'on pose $M = \text{ppcm}(1, \dots, M)$, on a :

$$a^M \equiv 1 \pmod{p} \text{ pour tout } a \text{ premier avec } p.$$

Autrement dit, p divise $a^M - 1$ et donc le pgcd de n et $a^M - 1$ est supérieur ou égal à p . En revanche, il est possible que le pgcd soit égal à n lui-même auquel cas, on n'obtient pas de facteur non trivial.

4.2 Pseudo-code

Choisir un résidu $x \pmod{n}$ au hasard et initialier p à 1 et un compteur cmp à 0. Définir une suite en posant $x_1 = x$, $x_2 = x_1^2 \pmod{n}$, $x_3 = x_2^2 \pmod{n}$, ... Ainsi x_{k+1} est obtenu en élevant x_k à la puissance $k+1$ modulo n . Autrement dit $x_k = x^{k!}$. Choisir une limite pour finir l'algorithme s'il ne trouve pas de résultats après un certain nombre d'essais.

4.3 Exemple

Nous factorisons le nombre 172189 avec notre algorithme $p-1$ de Pollard. On a $172189 = 409 \times 421$ et

$$409 - 1 = 408 = 2^3 \times 3 \times 17$$

puis

$$421 - 1 = 420 = 2^2 \times 3 \times 5 \times 7.$$

Voici ce que l'on obtient avec cet exemple :

k	1	2	3	4	5	6	7
$x_k = x^{k!} \pmod{n}$	2	4	64	74883	27019	147176	45890
$\text{pgcd}(x_k - 1, n)$	1	1	1	1	1	1	421

Au premier tour on trouve donc 421. On le fait ensuite sur $172189 \div 421 = 409$. Or avec un test de primalité, on voit directement que 409 est premier. Mais voici tout de même ce que l'on a en résultat :

Algorithme 1 : Factorisation de n par $p - 1$ de Pollard

Entrées : Un entier n **Sorties** : Un facteur premier p de n $x \pmod{n};$ $p = 1;$ $cmp = 0;$ $x_k = x^{k!};$ $lim;$ **Répéter** $x_k = x^{k!} \pmod{n};$ $p = \text{pgcd}(x_k - 1, n);$ $k = k + 1;$ $cmp = cmp + 1;$ **jusqu'à** $p \neq 1 \parallel cmp == lim;$ **if** $cmp == lim \ \&\& \ p == 1$ **then**| **Retourner** n **end****Retourner** p

k	1	2	3	4	5	6	...
$x_k = x^{k!} \pmod{n}$	2	4	64	36	25	345	...
$\text{pgcd}(x_k - 1, n)$	1	1	1	1	1	1	...

L'algorithme nous renvoie 409 car il ne trouve pas de facteur, c'est normal car il est premier.

4.4 Les limites

Dans certains cas, l'algorithme nous renvoie le même nombre mis en entrée ou une factorisation incomplète de celui-ci, en effet, cela correspond aux cas où les $p - 1$ ne sont pas ultrafriables.

Par exemple, avec le nombre 7345461, l'algorithme nous renvoie une factorisation incomplète. La factorisation naïve nous renvoie $7345461 = 3 \times 563 \times 4349$ tandis que $p - 1$ de Pollard nous renvoie $7345461 = 3 \times 2448487$. A priori, on suppose donc que 562 et 4348 ne sont pas ultrafriables, et en effet : $562 = 2 \times 281$ et $4348 = 2 \times 1087$.

5 Crible de Dixon

Le crible de Dixon se base sur la recherche de congruences de carrés. Son fonctionnement s'inspire de celui de l'algorithme de factorisation de Fermat qui consistait à écrire n comme la différence de deux carrés. On avait alors :

$$n = a^2 - b^2 = (a - b)(a + b).$$

5.1 Principe de l'algorithme

On cherche deux entiers a et b tels que $a^2 \equiv b^2 \pmod{n}$ et $a \not\equiv b \pmod{n}$. Un facteur de n pourra alors être trouvé en calculant $\text{pgcd}(a - b, n)$.

Pour cela, on choisit une borne B et on note k le nombre de premiers p inférieurs à B . On appelle P l'ensemble de ces premiers (de cardinal k , donc).

On prend ensuite un x aléatoirement dans $[1, n - 1]$ et on calcule $y \equiv x^2 \pmod{n}$. Si y est B -friable, on garde le couple (x, y) appelé *relation*. Appelons R l'ensemble des relations et m son cardinal. On recommence l'opération jusqu'à avoir $m > k$. On définit $v_{p,i}$ par $x_i^2 \pmod{n} = y_i = \prod_{p \in P} p^{v_{p,i}}$.

On construit ensuite la matrice $M = (v_{p,i} \pmod{2})_{p \in P, 1 \leq i \leq m}$, puis on trouve un vecteur non nul $(e_1, \dots, e_m)^t$ dans le noyau de M . On a alors :

$$\prod_{i=1}^m x_i^{2e_i} = \prod_{i=1}^m \prod_{p \in P} p^{v_{p,i} e_i} = \prod_{p \in P} p^{\sum_{i=1}^m v_{p,i} e_i} \pmod{n}.$$

Or (e_1, \dots, e_m) annule M , donc :

$$\sum_{i=1}^m v_{p,i} e_i \equiv 0 \pmod{2}.$$

On a donc une congruence de carrés :

$$a^2 = b^2 \pmod{n} \text{ avec } a = \prod_{i=1}^m x_i^{e_i} \text{ et } b = \prod_{p \in P} p^{\frac{1}{2} \sum_{i=1}^m v_{p,i} e_i}$$

qui nous donnera un facteur de n en calculant $\text{pgcd}(a - b, n)$.

5.2 Exemple

Factorisons $n = 7081$ avec $B = 3$. Considérons -1 comme un premier. Les entiers B -friables sont donc de la forme $\pm 2^k 3^l$, $k, l \in \mathbb{N}$.

On trouve trois carrés modulo n qui sont 3-friables :

$$\begin{aligned} 4486^2 &\equiv -2 \times 3 \pmod{n}, \\ 1857^2 &\equiv 2 \pmod{n}, \\ 2645^2 &\equiv -3 \pmod{n}. \end{aligned}$$

On construit alors

$$M = \begin{array}{c|ccc} & 4486 & 1857 & 2645 \\ \hline -1 & 1 & 0 & 1 \\ 2 & 1 & 1 & 0 \\ 3 & 1 & 0 & 1 \end{array}$$

Le vecteur colonne $(1, 1, 1)^t$ appartient au noyau de M , on a donc :

$$(4486 \times 1857 \times 2645)^2 \equiv (-2 \times 3)^2 \pmod{n}.$$

On calcule ensuite $\text{pgcd}(4486 \times 1857 \times 2645 - (-2 \times 3), 7081)$. On trouve 73 qui est donc un facteur non-trivial de 7081.

5.3 Pseudo-code

5.4 Précisions sur l'algorithme

5.4.1 Facteur non trivial

L'algorithme présenté retourne un facteur de n , éventuellement 1 ou n . Toutefois, si n est impair et le produit d'au moins deux premiers, le facteur trouvé est différent de 1 et n avec une probabilité supérieure à $1/2$.

En effet, supposons $n = pq$ avec p et q premiers. Posons $a^2 \equiv b^2 \pmod{n}$ où $a \in \llbracket 1, n-1 \rrbracket$. On a donc le tableau de probabilités suivant :

	$a \equiv b \pmod{p}$	$a \equiv -b \pmod{p}$
$a \equiv b \pmod{q}$	1/4	1/4
$a \equiv -b \pmod{q}$	1/4	1/4

Algorithme 2 : Factorisation de n par le crible de Dixon

Entrées : Un entier n et une borne B

Sorties : Un facteur de n , ou 1 ou n

$k = \#\{p \text{ premiers } | p < B\};$

$L = \text{liste_relations};$

for $i \in \{0, 1, \dots, k + 10\}$ **do**

$x \pmod{n};$

$y \equiv x^2 \pmod{n};$

if y B -friable **then**

$L[i] \leftarrow y$

end

end

$M = (v_{p,i} \pmod{2})_{p \in P, 1 \leq i \leq k+10}$ où $x_i^2 \pmod{n} = y_i = \prod_{p \in P} p^{v_{p,i}};$

(e_1, \dots, e_{k+10}) tel que $M(e_1, \dots, e_{k+10})^t = 0;$

$u = \prod_{i=1}^m x_i^{e_i};$

$v = \prod_{p \in P} p^{\frac{1}{2} \sum_{i=1}^m v_{p,i} e_i};$

Retourner $\text{pgcd}(u - v, n)$

On obtiendra donc un facteur non trivial de n dans la moitié des cas (lorsque $a \equiv b \pmod{p}$ et $a \equiv -b \pmod{q}$, ou $a \equiv -b \pmod{p}$ et $a \equiv b \pmod{q}$).

En fait, si n a m , $m \geq 2$, facteurs premiers, la probabilité d'obtenir un facteur non trivial est égale à $1 - \frac{1}{m^2} \geq \frac{1}{2}$ ($1 - \text{probabilité que } a \equiv b \pmod{p_1, p_2, \dots, p_m} \text{ ou que } a \equiv -b \pmod{p_1, p_2, \dots, p_m}$).

Dans les faits, on teste d'abord si n est une puissance exacte (avec la méthode de Newton généralement), puis s'il est premier avec un algorithme de primalité (Rabin-Miller, par exemple), et s'il n'est ni une puissance exacte, ni (vraisemblablement) pas premier, on peut alors le factoriser avec le crible de Dixon.

5.4.2 Choix de B

Le choix de B est crucial. En effet, si B est trop petit, on n'aura pas suffisamment de relations différentes pour trouver un vecteur non nul dans le noyau de M , et plus B augmente, plus l'algorithme mettra du temps à trouver des congruences entre carrés.

Lemme 3. *La probabilité qu'un entier n inférieur à un entier C soit B -friable est à peu près égale à u^{-u} , où $u = \frac{\log(C)}{\log(B)}$.*

Esquisse de preuve. Soit $\Psi(C, B) = \#\{x \in \llbracket 1, C \rrbracket; x \text{ est } B\text{-friable}\}$. La probabilité P qu'un entier n inférieur à C soit B -friable est donc égale à $\frac{\Psi(C, B)}{C}$.

Calculons maintenant P de manière approximative. On sait que $B^u = C$ et on définit donc

$$\begin{aligned} \Pi : \quad \llbracket 1, B \rrbracket^u &\rightarrow \llbracket 1, C \rrbracket \\ (x_1, x_2, \dots, x_u) &\mapsto \prod_{i=1}^u x_i \end{aligned}$$

L'image de Π est donc l'ensemble des entiers B -friables inférieurs à C . On a donc $\#(\text{Im } \Pi) = \Psi(C, B)$. Calculons alors $\#(\text{Im } \Pi)$ de manière approximative :

$$\#(\text{Im } \Pi) = \frac{\#(\llbracket 1, B \rrbracket^u)}{\#\{\text{éléments qui ont la même image}\}}$$

Or, $\#(\llbracket 1, B \rrbracket^u) = B^u$ et $\#\{\text{éléments qui ont la même image}\} \approx u!$ en admettant que le nombre d'éléments de $\llbracket 1, B \rrbracket^u$ qui ont la même image

est proche du nombre de permutations possibles dans cet ensemble. Ainsi, $\#(\text{Im } \Pi) = \frac{B^u}{u!} = \frac{C}{u!}$. On a donc :

$$P = \frac{\Psi(C, B)}{C} = \frac{\#(\text{Im } \Pi)}{C} \approx \frac{\frac{C}{u!}}{C} = \frac{1}{u!}$$

Lemme 4 (Formule de Stirling). $n! \approx \sqrt{2\pi n} \left(\frac{n}{e}\right)^n$ pour tout $n \in \mathbb{N}$.

La formule de Stirling peut aussi s'écrire $n! = n^n$ pour un grand entier n car $\frac{\sqrt{2\pi n}}{e^n}$ est alors négligeable devant n^n . On a donc $P \approx \frac{1}{u^u} \approx u^{-u}$. \square

On peut maintenant déterminer la valeur la plus adéquate de B pour le crible.

Lemme 5. La valeur optimale de B dans le crible de Dixon est $e\sqrt{\log(n)}$.

Démonstration. Le temps T que met l'algorithme est à peu près égal à

$$\text{nombre de relations} \times \frac{1}{\text{probabilité que } Y \text{ soit lisse}}.$$

Le nombre de relations est proche de B , et on appelle P la probabilité que Y soit lisse. On a donc :

$$\begin{aligned} T &= \frac{B}{P} \\ \log(T) &= \log(B) - \log(P) \\ &= \log(B) - \log(u^{-u}) \\ &= \log(B) + u \log(u) \\ &= \log(B) + \frac{\log(C)}{\log(B)} \times \log(u) \\ &= \log(B) + \frac{\log(n)}{\log(B)} \times \log(u) \end{aligned}$$

On étudie donc la fonction $f : x \mapsto x + \frac{\log(n)}{x}$, dont la dérivée est $f' : x \mapsto 1 - \frac{\log(n)}{x^2}$. f atteint son minimum en $x = \sqrt{\log(n)}$. On choisit alors $\log(B) = \sqrt{\log(n)}$, donc $B = e\sqrt{\log(n)}$. \square

Avec un tel B , on peut alors calculer T . On a donc $u = \frac{\log(n)}{\sqrt{\log(n)}} = \sqrt{\log(n)}$.

$$\begin{aligned}\log(T) &= \log(B) + \frac{\log(n)}{\log(B)} \times \log(u) \\ &= \sqrt{\log(n)} + \sqrt{\log(n)} \times \log(\sqrt{\log(n)})\end{aligned}$$

$\log(\sqrt{\log(n)})$ est négligeable devant $\sqrt{\log(n)}$, on a donc

$$\log(T) \approx 2\sqrt{\log(n)},$$

puis

$$T \approx e^{2\sqrt{\log(n)}}.$$

5.5 Limites de l'algorithme

Contrairement à l'algorithme $p-1$ de Pollard, la complexité dépend ici de n et plus du plus petit facteur premier de n . Le crible de Dixon est donc plus efficace pour factoriser des entiers de type $n = pq$ avec p et q premiers (comme dans RSA, par exemple) que pour trouver des petits facteurs de grands entiers.

6 Crible quadratique

L'algorithme du crible quadratique est un algorithme de factorisation fondé sur l'arithmétique modulaire, c'est-à-dire l'ensembles de méthodes, dérivées de l'étude du reste obtenu par une division euclidienne, permettant la résolution de problèmes sur les nombres entiers. En pratique c'est l'algorithme le plus rapide, sauf pour les nombres d'au moins cent chiffres décimaux, pour lesquels le crible algébrique est plus performant. Le temps d'exécution du crible quadratique dépend uniquement de la taille de l'entier à factoriser, et non de propriétés particulières de celui-ci.

6.1 Principe de l'algorithme

L'algorithme, mis au point en 1981 par Carl Pomerance, est un raffinement de la méthode de factorisation de Dixon. Le but est d'essayer d'établir

une congruence de carrés modulo n (l'entier à factoriser) qui nous permettra de factoriser n .

L'algorithme fonctionne en deux phases :

- la phase de collecte des données, où il collecte les informations qui peuvent conduire à une congruence de carrés, et
- la phase d'exploitation des données, où il place toutes les données qu'il a collectées dans une matrice et la résout pour obtenir une congruence de carrés.

Por la phase de collecte des données, on choisit un entier m proche de la racine carrée de n

$$m = \lfloor m^{\frac{1}{2}} \rfloor.$$

Ensuite on forme des congruences modulo n en observant que pour tout entier a ,

$$(m + a)^2 \equiv (m^2 - n) + a^2 + 2am \pmod{n},$$

où $m^2 - n$ est de l'ordre de \sqrt{n} . On se donne une borne B et l'on cherche des petits entiers a tels que $(m^2 - n) + a^2 + 2am \pmod{n}$ soit B -friable. À partir de la décomposition en facteurs premiers des nombres trouvés B -friable, on porte dans une matrice la parité des valuations et on forme des carrés à partir des lignes annulées par cette matrice. L'ensemble de ces lignes est un espace vectoriel. On trouve donc une congruence de carrés comme $b^2 = c^2 \pmod{n}$. Pour trouver un facteur p de n , on calcule $p = \text{pgcd}(b - c, n)$.

6.2 Pseudo-code

6.3 Exemple