Enjeux et avancées de la théorie algorithmique des nombres

François Morain *†‡

21 février 1992

1 Introduction

L'apparition des systèmes de chiffrement à clefs publiques de façon générale [DH76], et du système de chiffrement RSA en particulier [ARS78], a causé un regain d'intérêt pour la théorie des nombres et en particulier l'arithmétique dans ses aspects calculatoires. Pour répondre à des questions aussi simples que celles concernant la décomposition des nombres en facteurs premiers, il a fallu donner des réponses algorithmiques prenant en compte la faisabilité des calculs ainsi que le temps imparti pour donner une réponse satisfaisante. Cela a provoqué l'essor de la *théorie algorithmique des nombres*.

Cet exposé est destiné à mettre en lumière les progrès accomplis depuis une dizaine d'années dans les domaines de la primalité des entiers (comment peut-on prouver qu'un entier de quelques centaines de chiffres décimaux est premier) ; factorisation des entiers (quels sont les facteurs d'un nombre qui n'est pas premier) ; logarithme discret (comment calculer des logarithmes dans des groupes finis).

L'accent sera mis sur l'intéraction très étroite entre les théories mathématiques de haut niveau employées (courbes elliptique, corps de nombres, etc.) et l'utilisation de moyens de calcul nouveaux (machines parallèles, réseaux de stations de travail).

*LIX, Laboratoire d'Informatique de l'Ecole Polytechnique Ecole Polytechnique, 91128 Palaiseau Cedex, France

†Institut National de Recherche en Informatique et en Automatique (INRIA), Domaine de Voluceau, B. P. 105 78153 LE CHESNAY CEDEX (France).

‡Mis à disposition par la Délégation Générale pour l'Armement.

Contents

2 Problématique

2.1 Un peu d'arithmétique

2.1.1 Division euclidienne

L'arithmétique élémentaire se trouve au centre de la majorité des algorithmes de cryptographie à clefs publiques. Il est bon d'introduire certaines notations et rappeler quelques propriétés élémentaires.

Définition 2.1 *Soient a et b sont deux entiers. On dit que b divise a, ce que l'on note $b \mid a$ si et seulement s'il existe un entier c tel que $a = bc$.*

Rappelons que la *division euclidienne* de l'entier a par l'entier $b > 0$ fournit deux entiers q et r tels que

$$a = bq + r, 0 \leq r < |b|.$$

Cette propriété permet de calculer aisément le plus grand diviseur commun de deux entiers a et b, en remarquant que

$$\mathrm{pgcd}(a, b) = \mathrm{pgcd}(b, r).$$

Cela fournit un algorithme, dit *algorithme d'Euclide*, qui calcule très rapidement ce pgcd. En outre, il permet aussi de trouver rapidement les solutions de l'équation

$$au + bv = d$$

où $d = \mathrm{pgcd}(a, b)$ (algorithme d'Euclide étendu, ou algorithme de Bézout). Plus précisément, si a et b sont plus petits que N, alors le calcul de $\mathrm{pgcd}(a, b)$ nécessite au plus $2.5 \log N$ étapes.

2.1.2 Congruences

Définitions. Nous allons faire un usage constant des *congruences*.

Définition 2.2 *Soient a, b et N trois entiers. On dit que a* est congru à b modulo N *si et seulement si $N \mid a - b$. On note $a \equiv b \bmod N$.*

Ainsi :

$$13 \equiv 3 \bmod 10; \quad 16 \equiv 4 \bmod 6.$$

On s'intéresse alors à l'ensemble $\mathbf{Z}/N\mathbf{Z} = \{0, 1, \ldots, N-1\}$ qui est l'ensemble des restes possibles de la division euclidienne de a par N. Si $a = qN + r$, on a clairement $a \equiv r \bmod N$.

L'intérêt des congruences réside en ce fait que l'on peut additionner, soustraire, et multiplier des congruences sans difficulté (on dit que $\mathbf{Z}/N\mathbf{Z}$ est un anneau). Ainsi, pour connaître le reste de la division euclidienne de 12×15 par 7, on calculera

$$12 \times 15 \bmod 7 \equiv 5 \times 1 \equiv 5 \bmod 7.$$

Le seul problème qui se pose est la définition de l'inverse. On montre que a admet un inverse modulo N si et seulement si a est premier avec N. Si c'est le cas, on utilise l'algorithme d'Euclide pour déterminer b tel que

$$a \times b \equiv 1 \bmod N.$$

Un sous-ensemble de $\mathbf{Z}/N\mathbf{Z}$ important est précisément celui des éléments inversibles modulo N :

$$(\mathbf{Z}/N\mathbf{Z})^{\times} = \{a \in \mathbf{Z}/N\mathbf{Z}, \exists\, b, ab \equiv 1 \bmod N\}.$$

Cet ensemble forme un groupe, dont le cardinal est noté $\varphi(N)$ (ce nombre est appelé *indicateur d'Euler de* N). Si la factorisation de N s'écrit

$$N = p_1^{\alpha_1} \times \cdots p_k^{\alpha_k}$$

alors

$$\varphi(N) = N \prod_{i=1}^{k} (1 - 1/p_i).$$

Théorèmes de Fermat et d'Euler. Un des résultats que nous utiliserons constamment est le "petit théorème de Fermat".

Théorème 2.1 *Soit p un nombre premier et a un nombre premier avec p. Alors*

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

Sa généralisation est fondamentale pour RSA.

Théorème 2.2 *Soit N un entier et a un nombre premier avec N. Alors*

$$a^{\varphi(N)} \equiv 1 \bmod N.$$

Calcul de a^e. D'un point de vue pratique, nous allons être amenés à calculer $a^e \bmod N$ pour des entiers assez grands (quelques centaines de chiffres décimaux).

On calcule récursivement $a^e \bmod N$ en utilisant les formules :

$$a^{2f} \bmod N \equiv (a^f \bmod N)^2 \bmod N,$$

$$a^{2f+1} \bmod N \equiv (a^f \bmod N)^2 \times a \bmod N.$$

Ainsi :

$$a^{11} \equiv (a^5)^2 \times a \equiv ((a^2)^2 \times a)^2 \times a$$

nécessite 5 multiplications, au lieu de 10 si on calcule en séquence a^2, a^3, ..., a^{11}. On montre que le calcul de a^e peut se faire en $O(\log e)$ opérations élémentaires dans le groupe.

Théorème Chinois. Ce théorème permet de ramener l'étude des propriétés de l'anneau $\mathbf{Z}/N\mathbf{Z}$ avec N composé à celui des $\mathbf{Z}/p\mathbf{Z}$ avec p premier divisant N.

Supposons que a et b soient deux entiers premiers entre eux. Deux entiers x_a et x_b étant donnés, on cherche x tel que $x \equiv x_a \bmod a$ et $x \equiv x_b \bmod b$. Pour ce faire, on écrit que a et b sont premiers entre eux : il existe u et v tels que

$$au + bv = 1.$$

Alors, une solution du problème posé est

$$x = (au)x_b + (bv)x_a.$$

Ce genre de résultat permet entr'autres d'affirmer

$$x \equiv 1 \bmod a, x \equiv 1 \bmod b \Rightarrow x \equiv 1 \bmod (ab).$$

Comparaisons des propriétés de trois ensembles. Dans le tableau ci-dessous, on présente les différences de calcul essentielles entre le corps des complexes, $\mathbf{C}$, l'anneau $\mathbf{Z}/p\mathbf{Z}$ avec p premier et $\mathbf{Z}/N\mathbf{Z}$ avec N composé.

Problème	$\mathbf{C}$	$\mathbf{Z}/p\mathbf{Z}$	$\mathbf{Z}/N\mathbf{Z}$
$+, -, \times$	facile	facile	facile
$1/a$,	facile	facile	$(a, N) = 1$
a^e	facile	facile	facile
$a^x = y$	facile	?	?
$\sqrt{a}$	facile	alg. Shanks	?
$P(X) = 0$	facile	alg. Berlekamp	?

La notion de facilité doit être prise dans le sens : il existe un algorithme dont le temps de calcul est polynômial dans la taille des entrées. Notons aussi que les problèmes modulo N sont aussi faciles que ceux modulo p si on connaît la factorisation de N (en utilisant le théorème chinois).

2.2 RSA ; logarithme discret

2.2.1 RSA

Soit $N = p \times q$ un produit de deux nombres premiers impairs distincts. Soit e un nombre premier avec $\varphi(N) = (p-1) \times (q-1)$ et soit d l'inverse de e modulo $\varphi(N)$. Le système de chiffrement RSA [ARS78] fonctionne de la manière suivante :

- pour le chiffrement, le message clair est un entier de $\mathbf{Z}/N\mathbf{Z}$ et le message chiffré est

$$y \equiv x^e \bmod N;$$

- pour le déchiffrement

$$y^d \bmod N = x^{ed} \bmod N \equiv x \bmod N.$$

La validité de cet algorithme repose sur le fait que

$$ed = 1 + k\varphi(N)$$

et que l'on peut appliquer le théorème d'Euler pour trouver

$$x^{ed} \equiv x(x^{\varphi(N)})^k \equiv x \bmod N$$

(Note : ce raisonnement n'est valable que si x est premier avec N, mais le résultat reste valide dans le cas contraire).

Dans un tel schéma, le couple (N, e) est appelée clef publique et le nombre d (ou, de façon équivalente la factorisation $N = p \times q$) la clef secrète. La sécurité de ce système repose sur le fait que connaissant N et e, il est très difficile de déterminer d (ou p et q). Les seules attaques connues à ce jour sont soit la factorisation brutale de N, soit l'exploitation de faiblesses de construction des clefs e.

Il est clair que ce système est d'autant plus sûr que la fabrication de clefs (i.e., de nombres premiers p et q) est plus facile que la reconstitution des clefs secrètes. La suite de l'exposé s'étendra d'avantage sur ces problèmes.

2.2.2 Echange de clefs

Deux interlocuteurs, A et B, veulent correspondre à l'aide d'une clef commune K_{AB} générée à partir de leur clefs personnelles K_A et K_B. Un des protocoles possibles est le suivant. On suppose connus un nombre premier p et un entier a premier avec p. A (resp. B) choisit comme clef K_A (resp. K_B) la valeur

$$K_A \equiv a^x \bmod p \quad (resp. K_B \equiv a^y \bmod p)$$

où x (resp. y) est un entier que A (resp. B) garde secret. Les deux partenaires prennent alors pour clef commune

$$K_{AB} \equiv (K_A)^y \equiv (K_B)^x \bmod p.$$

Bien sûr, la sûreté de ce système repose sur l'impossibilité, connaissant a et $a^x \bmod p$, d'en déduire la valeur de x. Ce problème est appelé *problème du logarithme discret* et la suite de l'exposé jettera quelques lumières sur la résolution pratique de ce problème.

3 Premières solutions

3.1 Factorisation

3.1.1 Préambule

Passons en revue quelques-unes des méthodes qui permettent de trouver les facteurs premiers d'un entier. Précisons dès l'entrée quelques points importants.

La méthode la plus simple pour factoriser un nombre N consiste à le diviser par tous les nombres premiers plus petits que sa racine carrée. Ainsi

$$\sqrt{143} \approx 12$$

et 143 admet un diviseur premier parmi $\{2, 3, 5, 7, 11\}$. On trouve ainsi que 11 divise 143.

On peut se demander pourquoi cette méthode ne suffit pas pour factoriser tous les nombres. Avant d'en expliquer la raison, citons quelques résultats mathématiques utiles à connaître. Le premier est le théorème des nombres premiers.

Théorème 3.1 (Hadamard, de la Vallée Poussin, 1896) *Soit $\pi(x)$ le nombre de nombres premiers plus petits que x. Alors*

$$\pi(x) \sim \frac{x}{\log x}.$$

Plusieurs années auparavant, Tschebycheff avait montré

Théorème 3.2 *Il existe deux constantes C et $C' = 6C/5$ telles que*

$$C < \frac{\pi(x)}{x/\log x} < C',$$

avec

$$C = \log \frac{2^{1/2}3^{1/3}5^{1/5}}{30^{1/30}} = 0.921\ldots$$

et donc $C' = 1.1055\ldots$

Supposons maintenant que l'on veuille factoriser ainsi un nombre de 100 chiffres décimaux. Il faudrait le diviser par tous les nombres premiers plus petits que 10^{50} ce qui en ferait plus que 8×10^{47} (d'après le théorème précédent). Admettant maintenant pouvoir effectuer une division d'un nombre de 100 chiffres par un nombre de 50 chiffres en 1 nanoseconde (ce qui est optimiste, vu l'état actuel de la technologie), il faudrait 8×10^{38} secondes pour mener ce calcul à bien, ce qui fait 2×10^{32} années, ce qui est beaucoup plus que l'âge de l'univers ...

D'autres méthodes doivent être envisagées. Une question naturelle à se poser est la suivante : quels sont les nombres que l'on a factoriser dans la pratique. Une des grandes classes de nombres qui sert de tests et de comparaison pour toutes les méthodes nouvelles est celle du "Projet Cunningham" [BLS+88]. Ce sont des nombres de la forme

$$b^n \pm 1$$

avec b petit. Cela inclut les nombres de Mersenne et ceux de Fermat. Pour de multiples raisons, ce ne sont pas des nombres très représentatifs. Cette situation a récemment empiré du fait de l'apparition sur le marché du crible algébrique qui est particulièrement bien adapté à la factorisation de ces nombres. RSA, Inc. a lancé un challenge sur les nombres de partition ainsi que sur des nombres fabriqués spécialement pour être difficile à factoriser.

3.1.2 Trouver des petits facteurs

En pratique, on cherche les petits facteurs premiers d'un nombre jusqu'à une borne donnée B. On peut ensuite utiliser des méthodes probabilistes comme la méthode ρ de Pollard [Pol75, Bre80] pour trouver des facteurs de 8 à 10 chiffres.

3.1.3 Méthode $p-1$

Cette méthode repose sur le théorème de Fermat. Soit p un facteur inconnu de N. Soit a un entier premier avec N, donc avec p. Si on trouve un entier k tel que

$$p-1 \mid k!$$

alors

$$p \mid a^{k!}-1$$

car $p \mid a^{p-1}-1$. Comme p divise aussi N, il divise le pgcd de $b \equiv a^{k!}-1 \bmod p$ et de N.

En pratique, on augmente k jusqu'à ce que l'on trouve un facteur de N. Cette méthode ne marche bien sûr que si au moins un des facteurs premiers p de N est tel que $p-1$ est *friable* (en anglais : *smooth*), c'est-à-dire qu'il n'a que des petits facteurs premiers.

Exemple. Factorisons $N = 143$ avec la méthode $p-1$ et $a = 2$. On obtient

k	$b = 2^{k!} - 1 \bmod 143$	$\mathrm{pgcd}(b, 143)$
2	3	1
3	63	1
4	26	13

Cette méthode a le mérite d'être simple et facile à programmer. De multiples améliorations sont néanmoins possibles [Bre86, Mon87, MS90]. Le plus grand facteur trouvé à ce jour a 34 chiffres décimaux.

D'autre part, le principe de cette méthode sera utilisé dans d'autres algorithmes de factorisation qui utilisent d'autres groupes. Signalons dès à présent que des méthodes similaires existent et qui utlisent la factorisation de $p+1$ [Wil82] ainsi que celle de $\Phi_k(p)$ où $\Phi_k(X)$ est le k-ième polynôme cyclotomique [BS89].

3.1.4 Combinaisons de congruences

Présentation de la méthode ; algorithme de Dixon Ce sont les méthodes reines de la factorisation. Tout part d'une idée de Fermat. On peut factoriser un nombre N si on arrive à l'écrire sous la forme

$$N = X^2 - Y^2 = (X-Y) \times (X+Y).$$

Notre nombre fétiche, 143, est simple à factoriser :

$$143 = 12^2 - 1^2 = (12-1) \times (12+1).$$

Kraitchik a lui eu l'idée de chercher des nombres X et Y tels que :

$$X^2 \equiv Y^2 \bmod N$$

mais $X \not\equiv \pm Y \bmod N$. De cette façon, $\operatorname{pgcd}(X-Y, N)$ est un facteur de N.

Expliquons comment on peut tirer parti de cette idée de façon simple. On peut choisir *au hasard* des nombres x et calculer $f(x) = x^2 \bmod N$, puis factoriser $f(x)$. C'est la méthode de Dixon [Dix81].

Toujours avec $N = 143$, on trouve

x	$f(x)$	$factorisation$
15	82	2×41
17	3	3
18	38	2×19
19	75	3×5^2
20	114	$2 \times 3 \times 19$

On remarque maintenant que si on multiplie les lignes correspondant à $x = 17$ et $x = 19$, on obtient

$$(17 \times 19)^2 \equiv 3 \times (3 \times 5^2) \equiv (3 \times 5)^2 \bmod 143.$$

Prenant $X \equiv 17 \times 19 \equiv 37 \bmod 143$ et $Y = 15$, on trouve que $\operatorname{pgcd}(37 - 15, N) = 11$ et on a factorisé N.

L'idée est donc de choisir une base de nombres premiers $\mathcal{B} = \{p_1, \ldots, p_k\}$ avec laquelle on essaie de factoriser les nombres $x^2 \bmod N$. On obtient ainsi une collection de nombres

$$f(x) = \prod_{i=1}^{k} p_i^{\alpha_i(x)}$$

et on essaie de trouver un ensemble $\mathcal{X}$ tel que

$$F = \prod_{x \in \mathcal{X}} f(x)$$

soit un carré parfait *dans* $\mathbf{Z}$. On récrit cela comme

$$\prod_{i=1}^{k} p_i^{\sum_{x\in\mathcal{X}} \alpha_i(x)}.$$

Le nombre F est un carré si et seulement si pour tout i

$$\sum_{x \in \mathcal{X}} \alpha_i(x)$$

est un nombre pair ou encore :

$$\sum_{x \in \mathcal{X}} \alpha_i(x) \equiv 0 \bmod 2.$$

Partant d'un ensemble de factorisations, on se ramène à un problème d'algèbre linéaire. On doit trouver des relations de dépendance dans une matrice à coefficients dans $\mathbf{Z}/2\mathbf{Z}$.

Reprenons l'exemple précédent, en introduisant la matrice :

$$A = \begin{pmatrix} 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 \\ 1 & 1 & 0 & 1 \end{pmatrix}.$$

On lui adjoint la matrice triangulaire “compagnon” :

$$C = \begin{pmatrix} 1 & & & \\ 0 & 1 & & \\ 0 & 0 & 1 & \\ 0 & 0 & 0 & 1 \end{pmatrix}.$$

A chaque étape de la réduction, la matrice C va enregistrer les opérations effectuées sur la matrice. La ligne pivot correspondant à la première colonne est la deuxième. On doit ajouter cette ligne à toutes celles contenant un 1 dans la première colonne, c'est-à-dire ici à la quatrième. Pour mémoriser cette opération, on additionne en même temps la deuxième ligne de C à sa quatrième ligne :

$$\begin{pmatrix} 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{pmatrix} \begin{pmatrix} 1 & & & \\ 0 & 1 & & \\ 0 & 0 & 1 & \\ 0 & 1 & 0 & 1 \end{pmatrix}.$$

La ligne pivot pour la deuxième colonne est la première, ce qui nous donne :

$$\begin{pmatrix} 0 & 1 & 0 & 0 \\ x & x & x & x \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix} \begin{pmatrix} 1 & & & \\ 0 & 1 & & \\ 1 & 0 & 1 & \\ 1 & 1 & 0 & 1 \end{pmatrix}.$$

On lit maintenant les relations de dépendance sur la matrice C, en regard des lignes de A ne contenant que des 0. Par suite :

$$L_1 + L_3 = 0$$

et

$$L_1 + L_2 + L_4 = 0$$

sont des relations de dépendance pour A.

Il nous suffit de savoir que trouver des relations de dépendance dans une matrice $k \times k$ peut se faire en temps $O(k^r)$ avec $r \leq 3$ (des algorithmes meilleurs que le précédent sont décrits dans [Wie86, LO90, Cop91a, Cop91b]).

Pour résumer la méthode, on a construit des couples (u_x, w_x) vérifiant

$$u_x^2 \equiv w_x \bmod N,$$

et on factorise les w_x sur une base.

On peut montrer que cette méthode, dûe à Dixon [Dix81] a un temps de calcul

$$O(L(N)^2)$$

où

$$L(x) = \exp \sqrt{\log x \log \log x}.$$

Le nombre de nombres premiers est $L(N)^{1/2}$.

D'un point de vue pratique, cet algorithme n'est pas efficace car on doit factoriser des nombres de l'ordre de grandeur de N. La section suivante permet de pallier ce problème.

Crible quadratique L'idée [Pom82] est d'utiliser le mécanisme précédent avec un autre mode de génération des relations. On considère le polynôme

$$P(X) = (X + \lfloor\sqrt{N}\rfloor)^2 - N$$

où $\lfloor x \rfloor$ désigne le plus grand entier plus petit que le réel x. Avec les notations précédentes, on prend

$$u_x = x + \lfloor\sqrt{N}\rfloor, w_x = P(x)$$

avec l'avantage que cette fois

$$P(x) \simeq 2x\sqrt{N}$$

qui est beaucoup plus petit que N si on se restreint à des x petits.

Exemple. Tentons toujours de factoriser $N = 143$. Cette fois, afin de rendre les choses plus faciles, on choisit de multiplier N par 7 (c'est l'idée du multiplicateur) avec $7N \equiv 1 \bmod 8$. Le polynôme $P(X)$ vaut

$$P(X) = (X + 31)^2 - 1001.$$

Pour des raisons pratiques, choisissons $\mathcal{B} = \{-1, 2, 5, 17, 23\}$. Quelques valeurs bien choisies sont

x	$u(x)$	$w(x)$
-4	27	$-2^4 \times 17$
-2	29	$-2^5 \times 5$
0	31	$-2^3 \times 5$
1	32	23
6	37	$2^4 \times 23$
10	41	$2^3 \times 5 \times 17$

D'où l'on tire la relation de dépendance suivante :

$$(u(-4) \times u(-2) \times u(10))^2 \equiv (2^6 \times 5 \times 17)^2 \bmod 1001, \mathrm{pgcd}(71 - 238, 143) = 13.$$

D'un point de vue pratique, on ne factorise pas les valeurs de $P(x)$ au hasard. On remarque que :

$$p \mid P(x) \Leftrightarrow (x + \lfloor\sqrt{N}\rfloor)^2 \equiv N \bmod p$$

ce qui implique que N est un résidu quadratique modulo p. D'autre part, cette équation a deux racines x_p^+ et x_p^-. On a alors

$$p \mid w_x \Leftrightarrow x \equiv x_p^+ \textit{ ou } x \equiv x_p^- \bmod p,$$

ce qui conduit à la notion de crible. Supposons que l'on cherche à factoriser des nombres w_x avec x dans l'intervalle $[-M, M[$. Dans un tableau $T(x)$, on va stocker tous les nombres premiers p tels que $x \equiv x_p^\pm \bmod p$. A la fin du criblage, on regarde s'il y a beaucoup de ces nombres premiers dans $T(x)$. Si oui, on peut essayer de factoriser réellement w_x.

L'analyse de cette méthode conduit à un temps de calcul

$$O(L(N)^{\sqrt{9/8}})$$

et un nombre de nombres premiers $L(N)^{1/\sqrt{8}}$.

Le succès de cette méthode repose sur plusieurs choses. Tout d'abord, l'opération de base est l'addition de nombres en simple précision. De plus, le mécanisme de base est vectoriel (crible sur de grands intervalles). Une amélioration de la méthode a consisté à utiliser plusieurs polynômes (variante MPQS de Montgomery [Sil87]) ayant les mêmes propriétés que celles de $P(X)$. On peut alors envisager de distribuer les calculs sur plusieurs machines. Le premier à utiliser une réseau de stations de travail a été Silverman [Sil87, CS88] en mettant en place un protocole de type maître–esclave. Avec un réseau de 10 SUN 3, il fallait 90 heures de temps CPU pour factoriser un nombre de 70 chiffres. De nombreuses autres améliorations sont possibles [LM90, LM91].

Lenstra et Manasse [LM90] sont passés au régime supérieur en utilisant la transmission de données à l'aide du courrier électronique. Ainsi, avec l'aide d'une dizaine de sites de par le monde, ils ont été en mesure de factoriser pour la première fois un nombre de 100 chiffres décimaux avec PMPQS (le premier P pour la large prime variation) en octobre 1988. Le record actuel est toujours 116 chiffres avec la participation de quelques dizaines de centres de par le monde.

Enfin, notons que la barre des 100 chiffres a été franchie sur monoprocesseur par te Riele et al en Mars 1991 [tLW91].

3.2 Primalité

D'après ce qui précède, il nous faut savoir construire des nombres premiers de 100 chiffres décimaux.

3.2.1 Préambule : les limites du théorème de Fermat

On va, là encore, se servir du théorème de Fermat. Supposons que l'on veuille savoir si N est un nombre premier ou s'il est composé. On calcule

$$2^{N-1} \bmod N.$$

Si N est premier, alors cette quantité doit valoir 1. Par contre, si elle ne ne vaut pas 1, alors N est composé. Il est important de noter que cette condition est une condition *nécessaire* de primalité et qu'elle n'est pas suffisante. En effet

$$2^{340} \equiv 1 \bmod 341$$

mais $341 = 11 \times 31$. On dit alors que 341 est un nombre *pseudo-premier en base* 2. Bien pire, on pourrait imaginer de remplacer 2 par un autre nombre dans le test. Malheureusement, il existe des nombres, appelés *nombres de Carmichäel*, qui vérifient

$$\forall\, a, a^{N-1} \equiv 1 \bmod N.$$

Le plus petit de ces nombres est 561. On sait depuis très peu de temps qu'il existe une infinité de tels nombres [AGP92].

3.2.2 Nombres pseudopremiers forts ; algorithme de Miller-Rabin

On écrit $N-1 = 2^s t$ avec s entier et t impair. Le polynôme $X^{N-1}-1$ se factorise comme suit :

$$X^{N-1}-1 = (X^t - 1) \times (X^t + 1) \times (X^{2t}+1) \cdots \times (X^{2^{s-1}t}+1).$$

Soit a un entier. Si N est premier, il divise $a^{N-1}-1$, donc il divise l'un des membres de droite :

$$(MR_a) : a^t \equiv 1 \bmod N \text{ } ou \text{ } \exists j, 0 \leq j < s, a^{2^j t} \equiv -1 \bmod N.$$

Un nombre composé N qui passe le test (MR_a) est appelé *nombre pseudo-premier fort en base* a (ou ppf-a).

On sait qu'il existe une infinité de nombres ppf-a. On sait également qu'un nombre composé N est ppf-a pour au plus un quart des a possibles. L'algorithme de Miller-Rabin exploite ce fait. Si après avoir essayé k bases, on n'a pas trouvé de a tel que N n'est pas ppf-a, on déclare N premier avec une probabilité d'erreur inférieure à $(1/4)^k$. Ce raisonnement doit être pris avec des pincettes, mais l'algorithme donne de bons résultats en pratique.

3.2.3 Méthodes classiques : les réciproques du théorème de Fermat

Le théorème de Fermat n'admet pas de réciproque directe. Toutefois, on résout le problème de la manière suivante.

Théorème 3.3 *Soit g tel que*

$$g^{N-1} \equiv 1 \bmod N$$

et

$$\forall\, q \mid N-1, q \textit{ premier }, g^{(N-1)/q} \not\equiv 1 \bmod N.$$

Alors N est premier.

Remarquons que ce théorème donne un *certificat de primalité* pour l'entier N. En effet, pour convaincre quelqu'un que le nombre N est bien premier, il suffit de lui donner le générateur g et la factorisation de $N-1$: il n'y a plus qu'à calculer quelques exponentielles modulaires pour être d'accord ou non.

On peut améliorer ce théorème de plusieurs manières, dont on trouvera un résumé dans [BLS75]. Notons que l'on peut aussi utiliser les facteurs de $N+1$ et ceux de $\Phi_k(N)$.

Notons que ce résultat suffit à la construction de nombres premiers pour RSA, car on peut construire un nombre premier p en construisant en fait une factorisation $F = 2p_1^{\alpha_1} \cdots p_k^{\alpha_k}$ et en testant si $F+1$ est premier. Pour de plus amples informations, voir [Pla79, CQ82, Gor84, Mau90]. Pour pouvoir s'affranchir de cette construction, on doit recourir à l'utilisation d'algorithmes de primalité beaucoup plus puissants, que l'on citera par la suite.

3.3 Logarithme discret

Le problème peut se formaliser ainsi. Soit G un groupe fini et g un élément de G. Si a est un autre élément de G, on cherche s'il existe x tel que

$$a = g^x$$

dans G. Par analogie avec le cas réel, on posera $x = \log_g(a)$. Le premier exemple étudié est celui de $G = (\mathbf{Z}/p\mathbf{Z})^\times$.

3.3.1 Algorithme des pas de bébé et des pas de géant

Cet algorithme est dû à Shanks [Sha71]. On suppose que $|G| < t^2$ avec t entier. On cherche x tel que $g^x = a$ dans G. On écrit x sous la forme $x = x_0 + x_1 t$ avec $0 \leq x_i < t$. On calcule alors séparément la liste des g^{x_0} pour tous les x_0 possibles et la liste des $ag^{-x_1 t}$. On recherche alors les valeurs communes, c'est-à-dire

$$g^{x_0} = ag^{-x_1 t} \Leftrightarrow g^{x_0 + x_1 t} = a.$$

Cette recherche se fait efficacement par tri ou hachage. Le nombre x est trouvé en faisant approximativement $O(t) = O(\sqrt{|G|})$ opérations.

Des raffinements sont possibles, surtout dans les cas où on connaît exactement $|G|$, le cas le plus favorable étant quand on connaît la factorisation de $|G|$ (voir [PH78]).

A titre d'exemple, on se place dans le groupe $G = (\mathbf{Z}/131\mathbf{Z})^\times$ dont 2 est générateur. Cherchons à résoudre l'équation

$$2^x \equiv 15 \bmod 131.$$

On choisit $t = 12$ et on calcule les quantités :

x	g^x	ag^{-xt}
0	1	15
1	2	94
2	4	100
3	8	59
4	16	99
5	32	44
6	64	5
7	128	75
8	125	77
9	119	$\boxed{107}$
10	$\boxed{107}$	33
11	83	102

et on en déduit $2^{10} \equiv 15 \times 2^{-12\times 9}$ soit

$$2^{118} \equiv 15 \bmod p.$$

3.3.2 Méthode des indices

Exposons la méthode sur un exemple. Considérons le nombre premier $p = 131$ pour lequel 2 génère $G = (\mathbf{Z}/p\mathbf{Z})^\times$. Supposons que l'on veuille résoudre l'équation

$$2^x \equiv 15 \bmod 131.$$

On commence par générer des équations au hasard :

$$\begin{array}{lcrcl}
2^7 & \equiv & 128 & \equiv & -3 \bmod p, \\
2^8 & \equiv & 125 & \equiv & 5^3 \bmod p, \\
2^9 & \equiv & 119 & \equiv & 7 \times 17 \bmod p, \\
2^{12} & \equiv & 35 & \equiv & 5 \times 7 \bmod p, \\
2^{14} & \equiv & 9 & \equiv & 3^2 \bmod p.
\end{array}$$

On traduit ces relations en terme de logarithmes :

$$\left\{\begin{array}{rcl}
7 & \equiv & \log(-1) + \log(3) \\
8 & \equiv & 3\log(5) \\
9 & \equiv & \log(7) + \log(17) \\
12 & \equiv & \log(5) + \log(7) \\
14 & \equiv & 2\log(3).
\end{array}\right.$$

les congruences étant toutes modulo $p - 1$. De ces relations, on déduit que

$$\log(5) \equiv 8/3 \bmod (p-1) \equiv 46 \bmod p,$$

$$\log(3) \equiv 7 \bmod ((p-1)/2)$$

et on vérifie que $\log(3) = 7 + 130/2 = 72$.

Une fois ces précalculs effectués, on peut déterminer le logarithme du nombre a de la même façon si a est composé de petits facteurs premiers, ou, dans le cas général, si on peut trouver *une* équation du type

$$a2^u \equiv 3^y 5^z \bmod p.$$

Ici, $a = 15 = 3 \times 5$ et par suite :

$$\log(a) = \log(3) + \log(5) \bmod 130 = 118.$$

On montre (Cf. [Adl79]) que la première phase de l'algorithme a un coût $L(p)^{\sqrt{2}}$ et la deuxième $L(p)^{1/\sqrt{2}}$. Notons que la première phase doit être faite une fois pour toute, mais que la recherche d'*un* logarithme se fait avec la deuxième phase uniquement.

Cet algorithme a été amélioré dans [COS86]. Le coût de la première phase descend à $L(p)$ et celui de la deuxième à $L(p)^{1/2}$.

4 L'apport des mathématiques récentes

4.1 Formes quadratiques

Soit Δ un entier négatif congru à 0 ou 1 modulo 4. On considère l'ensemble des *formes quadratiques* $Q = (a, b, c)$, c'est-à-dire des fonctions

$$Q(x, y) = ax^2 + bxy + cy^2$$

à coefficients entiers tels que $b^2 - 4ac = \Delta$. Plus précisément, on va considérer uniquement les formes vérifiant en plus les conditions :

$$-a < b \leq a \leq c$$

et $a = c$ ou $a = |b|$ implique $b \geq 0$. Ces formes, dites *réduites* forment un ensemble fini, noté $\mathcal{H}(\Delta)$. On peut munir ce groupe d'une loi de composition, notée $\cdot$. Le cardinal de $\mathcal{H}(\Delta)$ est noté $h(\Delta)$.

Muni de ces informations, il est très facile d'imaginer une méthode de factorisation de type $p-1$, qui va marcher si et seulement si $h(-N)$ est friable [SL84]. Shanks a donné une méthode de factorisation basée sur les formes ambiges [Sha71, Coh72, Coh91]. A partir de cette méthode, on peut alors imaginer un algorithme de type combinaisons de congruences, dans laquelle on remplace la factorisation de nombres auxiliaires par celle de formes quadratiques [Sey87]. Signalons que cette dernière méthode a un temps de calcul heuristique de $L(N)^1$ (voir [LP91]).

D'autre part, il est aussi facile de concevoir que le problème du logarithme discret se pose dans le groupe $\mathcal{H}(\Delta)$. Un algorithme de type combinaisons de congruences a été proposé par McCurley [McC89].

4.2 Théorie algébrique des nombres

Un corps de nombre $K = \mathbf{Q}(\theta)$ est défini comme étant l'ensemble des nombres de la forme

$$\{x_0 + x_1\theta + \cdots + x_{n-1}\theta^{n-1}\}$$

où les x_i sont des nombres rationnels et θ un nombre complexe (ou réel) racine d'un polynôme à coefficients complexes :

$$a_n\theta^n + a_{n-1}\theta^{n-1} + \cdots + a_0 = 0.$$

L'exemple le plus simple est celui du corps $\mathbf{Q}(i)$, ensemble des nombres de la forme $\{a + bi\}$ avec a et b rationnels.

Dans de tels corps, on peut parler d'entiers et de factorisation d'entiers. Malheureusement, cette fois, la factorisation perd son caractère d'unicité. Il suffit de considérer pour cela le cas :

$$6 = 2 \times 3 = (1 + \sqrt{-5})(1 - \sqrt{-5})$$

où on obtient deux factorisations distinctes de l'entier 6. Pour pallier ce problème. on est conduit à introduire la théorie des idéaux de corps de nombres. Ces objets, un peu plus généraux que les nombres, forment un groupe et on peut alors factoriser de manière unique un idéal en produits d'idéaux premiers.

4.2.1 Factorisation

Malgré ces complications d'ordre théorique, on peut mettre au point des algorithmes de factorisation d'entiers utilisant des factorisations intermédiaires dans des corps de nombres. C'est le cas de l'algorithme NFS (*Number Field Sieve*) dû à Pollard [LLMP90, LLMP91]. Cet algorithme appartient également à la famille des algorithmes à combinaisons de congruences. A l'heure actuelle, il est très bien adapté à la factorisation des nombres de la forme $r^e \pm s$ avec r et s petits. Pour un nombre N de cette forme, on montre que le temps de calcul est

$$\exp(c(\log N)^{1/3}(\log\log N)^{2/3})$$

avec c une constante positive (effective). Cet algorithme a ainsi permis de factoriser le nombre $F_9 = 2^{2^9} + 1$, un nombre de 155 chiffres décimaux grâce à la collaboration d'une centaine de chercheurs de par le monde [LLMP91]. Cet algorithme ne permet pas, à l'heure actuelle, de factoriser des grands nombres de forme quelconque [BLP92].

4.2.2 Logarithme discret

Notons également que par un transfert courant, NFS peut s'adapter à la résolution du problème de logarithme discret dans $\mathbf{Z}/p\mathbf{Z}$ (Cf. [Gor92]) donnant un algorithme (théorique) de complexité analogue à celle citée ci-dessus pour NFS. Un algorithme pratique utilisant seulement les entiers de Gauss a été implanté par Odlyzko et LaMacchia [LO].

Le problème du log discret se pose aussi pour les (classes d') idéaux d'un corps de nombre. On pourra consulter [BD90] pour cela.

4.2.3 Sommes de Jacobi

Le premier algorithme de primalité tous-usages est basé sur des lois de réciprocité de haut degré et utilise les propriétés des corps cyclotomiques (corps $K = \mathbf{Q}(\zeta_n)$ où $\zeta_n = \exp(2i\pi/n)$. Il a été introduit par Adleman, Pomerance et Rumely [APR83] et perfectionné par Cohen et Lenstra [CL84] et implanté par Cohen et Lenstra [CL87, BvdH90]. Cet algorithme permet de prouver la primalité d'entiers de 200 chiffres en moins de quelques minutes de Cray. Son temps de calcul est

$$O((\log N)^{c \log\log\log N}).$$

4.3 Courbes elliptiques

Dans un corps k quelconque (de caractéristique différente de 2 et 3), on considère l'ensemble $E(k)$ des solutions de l'équation

$$y^2 = x^3 + ax + b \tag{1}$$

où a et b sont deux éléments de k tels que $-16(4a^3 + 27b^2)$ soit inversible dans k.

Plaçons nous dans $\mathbf{R}$. Si le deuxième membre de (1) a trois racines réelles, la courbe a l'allure donnée dans la figure 1. Ces courbes ont été très étudiées du fait qu'elles sont liées

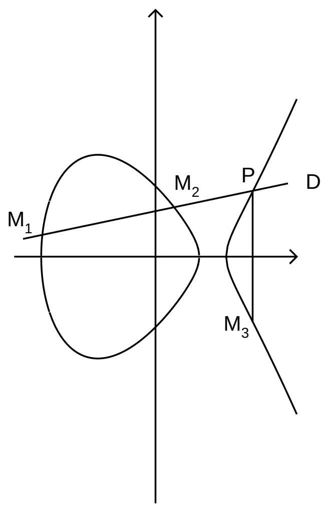


Figure 1: Une courbe elliptique sur $\mathbf{R}$.

au calcul des intégrales elliptiques du type

$$\int \frac{d\varphi}{\sqrt{1 - \lambda^2 \sin^2 \varphi}}$$

que l'on retrouve souvent en physique.

L'un des intérêts de considérer ces ensembles est que l'on peut définir une loi de groupe sur l'ensemble des points solutions de l'équation (1). On note habituellement cette loi $+$. Soient

$M_1 = (x_1, y_1)$ et $M_2 = (x_2, y_2)$ deux points de E. On veut calculer leur somme $M_3 = (x_3, y_3)$. On intersecte la droite M_1M_2 avec la courbe E et on voit que cette droite recoupe la courbe en un troisième point P. On prend alors comme point M_3 le symétrique de P par rapport à l'axe des x. Si $M_1 = M_2$, le raisonnement ci-dessus est valide en prenant la tangente en M_1 à E.

L'équation de la droite M_1M_2 se trouve sans difficulté avec sa pente, λ. On obtient

$$\begin{cases} x_3 &= \lambda^2 - x_1 - x_2 \\ y_3 &= \lambda(x_1 - x_3) - y_1 \end{cases}$$

où

$$\lambda = \begin{cases} (y_2 - y_1)(x_2 - x_1)^{-1} & si\ x_2 \neq x_1 \\ \\ (3x_1^2 + a)(2y_1)^{-1} & sinon. \end{cases}$$

On prend ordinairement ces équations comme définition de la loi de groupe dans un corps k quelconque.

Dans $\mathbf{Z}/p\mathbf{Z}$ ($p > 3$), une courbe elliptique a un nombre fini de points, noté m. Le théorème suivant permet de donner un encadrement de m.

Théorème 4.1 (Hasse)

$$m = p + 1 - t, |t| \leq 2\sqrt{p}.$$

Réciproquement, à tout nombre m_0 de l'intervalle $[(\sqrt{p} - 1)^2; (\sqrt{p} + 1)^2]$, on peut associer une courbe elliptique dont le nombre de points est m_0.

Il est maintenant facile de mettre au point une méthode de factorisation d'entiers de type $p - 1$ en utilisant la loi de groupe. Une courbe elliptique E étant fixée, cette méthode va donner un facteur p de N si le nombre de points de $E(\mathbf{Z}/p\mathbf{Z})$ est friable. La supériorité de cette méthode, dite *ECM* due à Lenstra [Len87], est qu'une courbe différente a un nombre de points différent. Ainsi, on a plus de chance de factoriser N en testant simultanément la friabilité de $p+1-t$ avec t variant. On peut prouver que le temps mis par ECM pour trouver un facteur p est $L(p)$, ce qui est beaucoup mieux que $L(N)$ dans le cas général.

Cette méthode est clairement parallélisable. De nombreuses astuces sont possibles [Mon87, AM91, Mon92]. Le plus grand facteur trouvé à ce jour a 40 chiffres décimaux.

Comme avec les groupes précédents, on peut envisager de résoudre le problème du logarithme discret. Celui-ci s'avère plus résistant que dans les autres cas [Mil87, Kob87a], car il n'y a pas de "factorisation" sur les courbes elliptiques. Notons que ce problème se résout dans certains cas au cas du logarithme discret dans un corps fini.

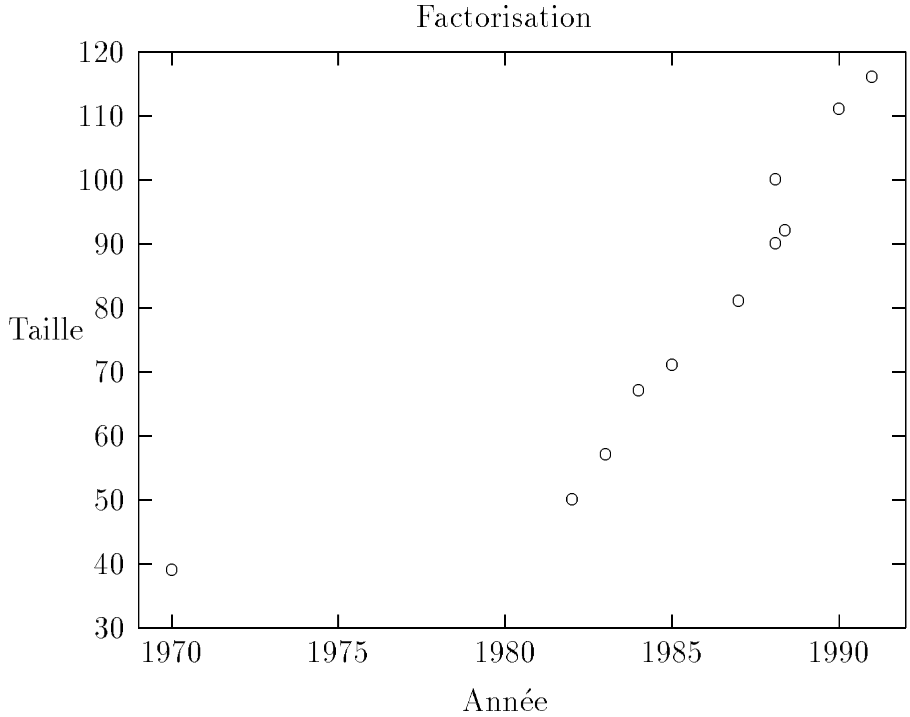
Enfin, l'introduction des courbes elliptiques a permis de donner un algorithme de primalité dont le temps de calcul est en moyenne polynômial. C'est l'œuvre de Goldwasser et Kilian (utilisant un algorithme de Schoof, voir [GK86]) sur le plan théorique et d'Atkin sur le plan pratique [AM90, Mor90]. L'algorithme, appelé ECPP (*Elliptic Curve Primality Proving* algorithm), permet de tester la primalité de nombres de 10 à 1200 chiffres décimaux en un temps raisonnable (sur réseaux de stations de travail). Cet algorithme requiert une théorie très importante (théorie algébrique des nombres, théorie de Galois, corps de classe) mais il marche très bien sur n'importe quel station de travail (deux minutes de temps CPU pour prouver la primalité d'un nombre de 100 chiffres décimaux sur une Dec Station 3100). Une version portable est disponible pour toute machine UNIX.

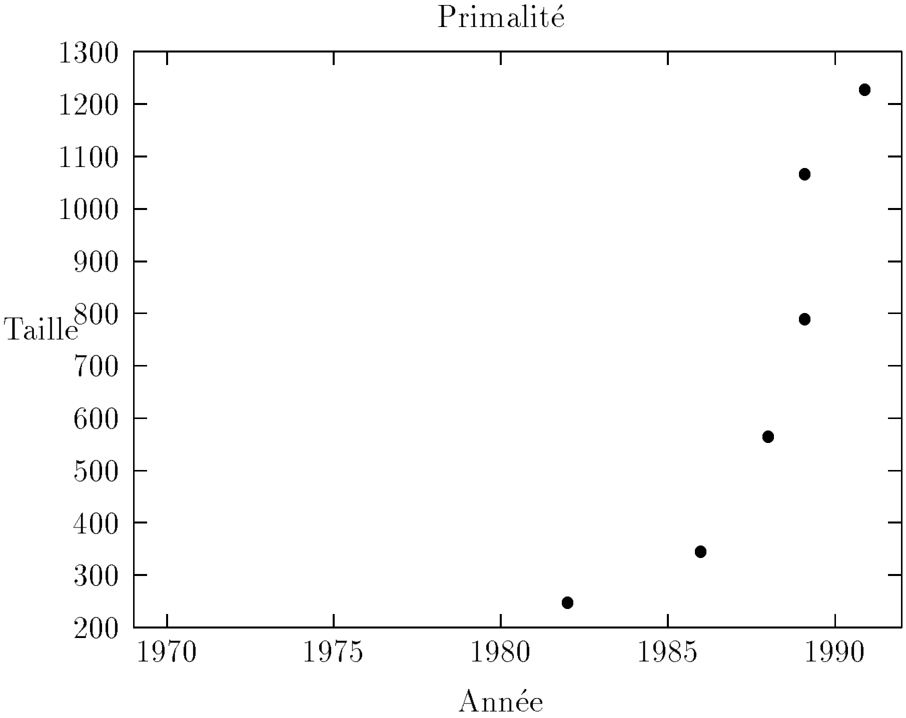
5 Conclusion : état de l'art

Terminons par un bref récapitulatif des avancées dans les trois domaines que nous avons passé en revue :

Problème	routine	record	méthode
Primalité	800 dig.	1226	ECPP
Factorisation	96 dig.	116	PPMPQS
	90 dig.	155	NFS
Log discret	–	60	Gaussian integers

A titre de curiosité, on trace sur les courbes suivantes l'évolution des deux records principaux au fil des ans. On constate aisément que le problème de la primalité des entiers a un ordre de grandeur d'avance sur celui de la factorisation.





Remarques bibliographiques. Parmi les ouvrages de base sur la théorie algorithmique des nombres, citons [Knu81]. Un ouvrage plus avancé est [Coh91]. Pour la théorie appliquée à la cryptographie à clefs publiques, voir [Rie85, Kob87b].

References

[Adl79] Adleman (L. M.). – A subexponential algorithm for the discrete logarithm problem with applications to cryptography. *In : FOCS.*

[AGP92] Alford (W. R.), Granville (Andrew) et Pomerance (Carl). – There are infinitely many carmichael numbers. – February 1992. In preparation.

[AM90] Atkin (A. O. L.) et Morain (F.). – *Elliptic curves and primality proving.* – Research Report n° 1256, INRIA, Juin 1990. Submitted to *Math. Comp.*

[AM91] Atkin (A. O. L.) et Morain (F.). – Finding suitable curves for the elliptic curve method of factorization. – March 1991. To appear in *Mathematics of Computation.* Also available as INRIA Research Report no 1547.

[APR83] Adleman (L. M.), Pomerance (C.) et Rumely (R. S.). – On distinguishing prime numbers from composite numbers. *Annals of Math.*, vol. 117, 1983, pp. 173–206.

[ARS78] Adleman (L. M.), Rivest (R. L.) et Shamir (A.). – A method for obtaining digital signatures and public-key cryptosystems. *Comm. ACM*, vol. 21, n° 2, 1978, pp. 120–126.

[BD90] Buchmann (Johannes) et Dullmann (Stephan). – On the computation of discrete logarithms in class groups. *In: Advances in Cryptology*, éd. par Menezes (A. J.) et Vanstone (S. A.). pp. 134–139. – Springer-Verlag. Proc. Crypto '90, Santa Barbara, August 11–15, 1988.

[BLP92] Buhler (J.), Lenstra, Jr. (H. W.) et Pomerance (Carl). – Factoring integers with the number field sieve. – 1992. Preprint.

[BLS75] Brillhart (J.), Lehmer (D. H.) et Selfridge (J. L.). – New primality criteria and factorizations of $2^m \pm 1$. *Math. Comp.*, vol. 29, n° 130, 1975, pp. 620–647.

[BLS+88] Brillhart (J.), Lehmer (D. H.), Selfridge (J. L.), Tuckerman (B.) et Wagstaff, Jr. (S. S.). – *Factorizations of* $b^n \pm 1$, $b = 2, 3, 5, 6, 7, 10, 11, 12$ *up to high powers.* – AMS, 1988, 2 édition, *Contemporary Mathematics.*

[Bre80] Brent (R. P.). – An improved Monte Carlo factorization algorithm. *BIT*, vol. 20, 1980, pp. 176–184.

[Bre86] Brent (R. P.). – Some integer factorization algorithms using elliptic curves. *In: Proc. 9th Australian Computer Science Conference.*

[BS89] Bach (Eric) et Shallit (Jeffrey). – Factoring with cyclotomic polynomials. *Math. Comp.*, vol. 52, n° 185, January 1989, pp. 201–219.

[BvdH90] Bosma (Wieb) et van der Hulst (Marc-Paul). – *Primality proving with cyclotomy.* – Thèse de PhD, Universiteit van Amsterdam, décembre 1990.

[CL84] Cohen (H.) et Lenstra, Jr. (H. W.). – Primality testing and Jacobi sums. *Math. Comp.*, vol. 42, n° 165, 1984, pp. 297–330.

[CL87] Cohen (H.) et Lenstra (A. K.). – Implementation of a new primality test. *Math. Comp.*, vol. 48, n° 177, 1987, pp. 103–121.

[Coh72] Cohen (Henri). – Méthodes de factorisation et nombre de classes des corps quadratiques. *In: Séminaire de Théorie des Nombres Delange–Pisot–Poitou.*

[Coh91] Cohen (Henri). – *A course in algorithmic algebraic number theory.* – Springer–Verlag, 1991, *Graduate Texts in Mathematics.* To appear.

[Cop91a] Coppersmith (Don). – *Solving linear equations over* $GF(2)$. – RC n° 16997, IBM Research, T. J. Watson Research Center, July 1 1991. Updated version of August 6, 1991.

[Cop91b] Coppersmith (Don). – Solving linear equations over $gf(2)$ via block Wiedemann algorithm. – 1991. Submitted to *Math. Comp.*

[COS86] Coppersmith (D.), Odlyzko (A. M.) et Schroeppel (R.). – Discrete logarithms in GF(p). *Algorithmica*, vol. 1, 1986, pp. 1–15.

[CQ82] Couvreur (C.) et Quisquater (J.J.). – An introduction to fast generation of large prime numbers. *Philips J. Research*, vol. 37, 1982, pp. 231–264.

[CS88] Caron (Thomas R.) et Silverman (Robert D.). – Parallel implementation of the quadratic sieve. *The Journal of Supercomputing*, vol. 1, 1988, pp. 273–290.

[DH76] Diffie (W.) et Hellman (M. E.). – New directions in cryptography. *IEEE Trans. on Information Theory*, vol. IT-22-6, nov 1976.

[Dix81] Dixon (J. D.). – Asymptotically fast factorization of integers. *Math. Comp.*, vol. 36, 1981, pp. 255–260.

[GK86] Goldwasser (S.) et Kilian (J.). – Almost all primes can be quickly certified. *In : Proc. 18th STOC*, pp. 316–329. – Berkeley, May 28–30 1986.

[Gor84] Gordon (D.). – Strong primes are easy to find. *In : Proc. Eurocrypt '84*. pp. 216–223. – Springer.

[Gor92] Gordon (Dan M.). – Discrete logarithms in $GF(p)$ using the number field sieve. – 1992. To appear in *SIAM Journal on Discrete Math.*

[Knu81] Knuth (D. E.). – *The Art of Computer Programming: Seminumerical Algorithms.* – Addison-Wesley, 1981.

[Kob87a] Koblitz (N.). – Elliptic curve cryptosystems. *Math. Comp.*, vol. 48, n° 177, January 1987, pp. 203–209.

[Kob87b] Koblitz (Neal). – *A course in number theory and cryptography.* – Springer–Verlag, 1987, *Graduate Texts in Mathematics*, volume 114.

[Len87] Lenstra, Jr. (Hendrik W.). – Factoring integers with elliptic curves. *Annals of Math.*, vol. 126, 1987, pp. 649–673.

[LLMP90] Lenstra (A. K.), Lenstra, Jr. (H. W.), Manasse (M. S.) et Pollard (J. M.). – The number field sieve. *In : Proceedings 22nd Annual ACM Symposium on Theory of Computing (STOC).*

[LLMP91] Lenstra (A. K.), Lenstra, Jr. (H. W.), Manasse (M. S.) et Pollard (J. M.). – The factorization of the ninth Fermat number. – 1991. To appear.

[LM90] Lenstra (A. K.) et Manasse (Mark S.). – Factoring by electronic mail. *In : Advances in Cryptology*, éd. par Quisquater (J.-J.). pp. 355–371. – Springer-Verlag. Proc. Eurocrypt '89, Houthalen, April 10–13.

[LM91] Lenstra (A. K.) et Manasse (M. S.). – Factoring with two large primes. *In : Advances in Cryptology – EUROCRYPT '90*, éd. par I. B. Damgård. pp. 72–82. – Springer–Verlag. Proceedings of the Workshop on the Theory and Application of Cryptographic Techniques, Aarhus, Denmark, May 21–24, 1990.

[LO] LaMacchia (B. A.) et Odlyzko (A. M.). – Computation of discrete logarithms in prime fields. – Preprint.

[LO90] LaMacchia (B. A.) et Odlyzko (A. M.). – Solving large sparse linear systems over finite fields. *In : Advances in Cryptology*, éd. par Menezes (A. J.) et Vanstone (S. A.). pp. 109–133. – Springer-Verlag. Proc. Crypto '90, Santa Barbara, August 11–15, 1988.

[LP91] Lenstra, Jr. (H. W.) et Pomerance (Carl). – A rigorous time bound for factoring integers. – February 1991. Preprint.

[Mau90] Maurer (U. M.). – Fast generation of secure RSA-products with almost maximal diversity. *In : Advances in Cryptology*, éd. par Quisquater (J.-J.). pp. 636–647. – Springer-Verlag. Proc. Eurocrypt '89, Houthalen, April 10–13.

[McC89] McCurley (K. S.). – Cryptographic key distribution and computation in class groups. *In : Proc. of NATO ASI Number Theory and Applications, Banff, May 1988*. pp. 459–479. – Kluwer Academic Publishers.

[Mil87] Miller (V.). – Use of elliptic curves in cryptography. *In : Advances in Cryptology*, éd. par Odlyzko (A. M.). pp. 417–426. – Springer-Verlag. Proceedings Crypto '86, Santa Barbara (USA), August11–15, 1986.

[Mon87] Montgomery (P. L.). – Speeding the Pollard and elliptic curve methods of factorization. *Math. Comp.*, vol. 48, n° 177, January 1987, pp. 243–264.

[Mon92] Montgomery (Peter L.). – Elliptic curves with torsion groups of orders 12 and 16 over q. – January 1992. Preprint.

[Mor90] Morain (François). – *Courbes elliptiques et tests de primalité.* – Thèse de PhD, Université Claude Bernard–Lyon I, Septembre 1990.

[MS90] Montgomery (P. L.) et Silverman (B.). – An FFT extension to the $P-1$ factoring algorithm. *Math. Comp.*, vol. 54, n° 190, April 1990, pp. 839–854.

[PH78] Pohlig (S.) et Hellman (M.). – An improved algorithm for computing logarithms over gf(p) and its cryptographic significance. *IEEE Transactions on Information Theory*, vol. 24, 1978, pp. 106–110.

[Pla79] Plaisted (D. A.). – Fast verification, testing and generation of large primes. *Theoretical Computer Science*, vol. 9, 1979, pp. 1–16.

[Pol75] Pollard (J. M.). – A Monte-Carlo method for factorization. *BIT*, vol. 15, 1975, pp. 331–334.

[Pom82] Pomerance (Carl). – Analysis and comparison of some integer factoring algorithms. *In : Computational methods in number theory*, éd. par Lenstra, Jr. (H. W.) et Tijdeman (R.). Mathematisch Centrum, Amsterdam, pp. 89–140. – Mathematical Center Tracts 154/155.

[Rie85] Riesel (Hans). – *Prime numbers and computer methods for factorization.* – Birkhäuser, 1985, 2nd édition, *Progress in Mathematics*, volume 57.

[Sey87] Seysen (M.). – A probabilistic factorization algorithm with quadratic forms of negative discriminant. *Math. Comp.*, vol. 48, n° 178, April 1987, pp. 757–780.

[Sha71] Shanks (D.). – Class number, a theory of factorization, and genera. *In : Proc. Symp. Pure Math. vol. 20.* AMS, pp. 415–440.

[Sil87] Silverman (R. D.). – Parallel implementation of the quadratic sieve. *The Journal of Supercomputing*, vol. 1, n° 3, 1987.

[SL84] Schnorr (C. P.) et Lenstra, Jr. (H. W.). – A Monte-Carlo factoring algorithm with linear storage. *Math. Comp.*, vol. 43, n° 167, July 1984, pp. 289–311.

[tLW91] te Riele (Herman), Lioen (Walter) et Winter (Dik). – Factorization beyond the googol with MPQS on a single computer. – March 1991. To appear.

[Wie86] Wiedemann (D. H.). – Solving sparse linear equations over finite fields. *IEEE Transac. Information Theory*, vol. **IT**-32, 1986, pp. 54–62.

[Wil82] Williams (H. C.). – A $p+1$ method of factoring. *Math. Comp.*, vol. 39, n° 159, July 1982, pp. 225–234.