Development of Mathematics 1950–2000

Edited by Jean-Paul Pier

Development of mathematics 1950–2000 / ed. by Jean-Paul Pier.
- Basel ; Boston ; Berlin : Birkhäuser, 2000
ISBN 3-7643-6280-4

Arithmétique et Cryptographie*

Jean-Louis Nicolas

Université Claude Bernard Lyon

1 Introduction

La deuxième moitié du XXème siècle a vu le développement rapide des ordinateurs. L'influence sur la théorie des nombres a d'abord été timide, et réservée à quelques spécialistes comme D.H. Lehmer et D. Shanks. Puis elle s'est accrue considérablement, et l'on peut dire maintenant que "l'utilisation des ordinateurs en théorie des nombres" (computational number theory) est devenue une spécialité à part entière dans les mathématiques.

La première méthode de factorisation vraiment nouvelle a été présentée par D. Shanks en 1969 (cf. [25]). Le problème initial de Shanks était de calculer le nombre de classes de formes quadratiques de discriminant négatif Δ donné assez grand en valeur absolue, et il s'était aperçu que la méthode fournissait en plus la factorisation de Δ. Une autre étape importante a été la publication en 1978 du protocole de cryptographie RSA [1] qui utilise la propriété suivante: il est possible de construire deux grands nombres premiers p et q (disons de 100 chiffres décimaux), et de calculer le produit $n = p \times q$; mais il n'existe pas actuellement de méthode efficace pour calculer p et q si l'on connaît seulement n. L'importance dans la vie actuelle de la sécurité informatique, et de la transmission secrète des données a encouragé considérablement les recherches autour de la méthode RSA, en particulier dans le domaine des tests de primalité et des méthodes de factorisation.

L'apport mathématique de ces problèmes est loin d'être négligeable. Par exemple, certains groupes finis jouent un rôle important dans ces sujets, essentiellement les groupes $(\mathbb{Z}/N\mathbb{Z})^*$, formés des entiers inversibles modulo N (ce qui est assez naturel), les groupes de classes de formes quadratiques de discriminant fixé longuement étudiés par Gauss, et les groupes de points des courbes elliptiques modulo un nombre premier p. Mais d'autres groupes classiques, par exemple les groupes multiplicatifs de matrices n'ont encore jamais pu être utilisés. Y a-t-il à cela une raison profonde? Un autre exemple est la fonction $\Psi(x, y)$ qui compte le nombre d'entiers positifs $\leq x$ dont tous les facteurs premiers sont $\leq y$. Cette fonction a été étudiée par de Bruijn avant 1966 (cf. [7]), et il se trouve qu'elle joue un rôle important dans toutes les méthodes efficaces actuellement connues de factorisation des nombres entiers pour estimer leur coût. Le troisième exemple est lié directement à la méthode RSA. S'il était aussi rapide de factoriser un nombre entier que de tester s'il est premier, le protocole RSA perdrait son intérêt. C'est un problème théorique important de

*AMS classification 11A51 11T71 11Y05 11Y11 94A60

[1] D'après les initiales de ses trois auteurs, Rivest, Shamir et Adleman cf. [2].

savoir si la complexité des méthodes de factorisation est comparable à celle des tests de primalité. Enfin, plusieurs protocoles de cryptographie ont été cassés, par exemple celui du sac à dos (cf. ci-dessous 5(b)). Est-il possible de prouver qu'un protocole est sûr? C'est là un problème de logique dont la réponse n'est pas évidente.

Dans le paragraphe suivant, nous présentons les *algorithmes de base* de la théorie des nombres. Ces algorithmes sont presque tous très anciens, mais ils ont été considérablement remis à l'honneur avec l'arrivée des ordinateurs, et ils constituent la bibliothèque de base pour les algorithmes présentés ultérieurement. Le paragraphe 3 présentera les *courbes elliptiques* qui ont joué un grand rôle dans le développement des mathématiques ces dernières années. Dans le paragraphe 4, nous parlerons des *tests de primalité*. Dans le paragraphe 5, nous décrirons quelques *protocoles de cryptographie*, les *méthodes de factorisation* seront présentées dans le paragraphe 6, et le calcul du *logarithme discret* dans le paragraphe 7. Dans le paragraphe 8, quelques *problèmes d'utilisation intensive d'ordinateurs* en mathématiques seront abordés.

2 Les algorithmes de base

On trouvera une description précise des algorithmes ci-dessous dans [9] ou [6].

(a) *La multiprécision.* L'arithmétique standard des ordinateurs opère sur des nombres d'une dizaine de chiffres décimaux. Il faut donc l'adapter à travailler sur des nombres plus grands. Les systèmes de calcul formel (MAPLE, MATHEMATICA,...) travaillent sur des entiers de longueur arbitraire, mais de façon assez lente. La librairie GMP de GNU ou celle de PARI[2] sont des bibliothèques de multiprécision plus rapide.

(b) *L'algorithme d'Euclide.* On notera a div b et a mod b le quotient et le reste dans la division de a par b. L'algorithme d'Euclide pour le calcul du pgcd de a et b est basé sur le fait que si $a = bq + r$, $\text{pgcd}(a, b) = \text{pgcd}(b, r)$. On obtient la forme récursive:

fonction Euclide (a, b)

si b=0 *alors retourner* a *sinon retourner* Euclide (b, a mod b) *finsi;*

fin.

Lamé en 1845 a démontré que le nombre de divisions dans l'algorithme d'Euclide était inférieur à 5 fois le nombre de chiffres décimaux de a ou de b. On voit ainsi que si a et b ont au plus 100 chiffres décimaux, le calcul de leur pgcd nécessite au plus 500 divisions, ce qui s'effectue très vite sur les ordinateurs actuels.

Il est facile d'adapter l'algorithme d'Euclide pour le calcul des coefficients de Bezout u et v tels que $au + bv = d = \text{pgcd}\,(a, b)$. Une forme récursive s'écrit simplement en remarquant que $a = bq + r$ et $bu' + rv' = d$ entraînent $au + bv = d$ avec

[2]que vous pourrez trouver sur le web aux adresses ftp://ftp.ibp.fr/pub/gnu/gmp*.tar.gz (GMP veut dire Gnu Multi-Précision) et ftp://megrez.ceremab.u-bordeaux.fr/pub/pari

$u = v'$ et $v = u' - qv'$. On utilise une liste $\ell = [u, v, d]$. Le i ème terme de la liste est désigné par $\ell[i]$:

fonction Euclide_étendu(a, b);

si b=0 *alors retourner* [1, 0, a];

sinon q:=a div b; r=a mod b;

l:=Euclide_étendu(b, r);

retourner [l[2],l[1]-q*l[2],l[3]];

finsi;

fin.

On utilise en général une version non récursive, un peu plus difficile à écrire, mais plus rapide à exécuter. Notons que, si a et b sont premiers entre eux, cet algorithme fournit $u = a^{-1}$, l'inverse de a modulo b.

(c) *Le théorème des restes des chinois.* Soit $n_1, n_2, \ldots, n_k$ des nombres premiers entre eux deux à deux et $a_1, a_2, \ldots, a_k$ des entiers quelconques. Soit le système

$$(\mathcal{S}) \begin{cases} x \equiv a_1 \bmod n_1 \\ x \equiv a_2 \bmod n_2 \\ \quad\vdots \\ x \equiv a_k \bmod n_k. \end{cases}$$

Il existe A, $0 \le A \le N-1$ tel que $\mathcal{S}$ soit équivalent à $x \equiv A \pmod N$, avec $N = n_1 n_2 \cdots n_k$. Pour calculer A, on peut procéder de la façon suivante: on pose $N_i = N/n_i$, n_i et N_i sont premiers entre eux. On leur applique l'algorithme d'Euclide étendu qui fournit u_i et v_i tels que $u_i n_i + v_i N_i = 1$. Alors,

$$A = a_1 v_1 N_1 + a_2 v_2 N_2 + \cdots + a_k v_k N_k \bmod N.$$

(d) *L'algorithme des puissances.* On veut calculer a^b. Si b est pair, on écrit $b = 2b'$ et $a^b = (a^{b'})^2$. Si b est impair, $b = 2b' + 1$, et $a^b = a(a^{b'})^2$. Cela fournit un algorithme récursif pour calculer a^b. On le transforme aisément en algorithme non récursif en écrivant b en base 2, en rayant le chiffre 1 sur la gauche et en remplaçant chaque chiffre 0 par Q et chaque chiffre 1 par QM. On obtient ainsi pour $b = 13$, $b = (1101)_2$: QMQQM. Pour calculer a^b, on exécute de la gauche vers la droite en partant de a les opérations $Q(t) = t^2$ et $M(t) = a \times t$. On obtient successivement a^2, a^3, a^6, a^{12} et a^{13}. D'après [9], cette méthode était connue en Inde 200 ans avant J.-C.

Le nombre de multiplications à effectuer est inférieur à deux fois le nombre de chiffres de b en base 2 diminuée de 1 soit au plus $2\dfrac{\log b}{\log 2}$.

Pour calculer $a^b \bmod N$, on procède de même. Simplement, pour éviter de manipuler de trop grands nombres, on pose $Q(t) = t^2 \bmod N$ et $M(t) = at \bmod N$. Si a, b, N sont des entiers d'au plus 100 chiffres, on peut calculer $a^b \bmod N$ en exécutant moins de $\dfrac{200}{\log_{10} 2} \leq 665$ multiplications de nombres de moins de 100 chiffres suivies d'une division par N, ce qui est très rapide.

La même méthode s'applique dans n'importe quel groupe: par exemple, a peut être une matrice. Elle est utilisée dans le groupe des classes de formes quadratiques, ou dans le groupe des points d'une courbe elliptique pour les méthodes de factorisation.

(e) *Les symboles de Legendre et de Jacobi.* Soit p un nombre premier $\neq 2$. Le symbole de Legendre $\left(\frac{a}{p}\right)$ est égal à $+1$ si a est un carré modulo p, à -1 si a n'est pas un carré et à 0 si a est multiple de p, ou encore $\left(\frac{a}{p}\right) \equiv a^{\frac{p-1}{2}} \pmod p$.

Si n est impair positif et $n = p_1 p_2 \cdots p_r$ avec $p_1 \leq p_2 \leq \cdots \leq p_r$, on définit le symbole de Jacobi

$$\left(\frac{a}{n}\right) = \left(\frac{a}{p_1}\right)\left(\frac{a}{p_2}\right)\cdots\left(\frac{a}{p_r}\right).$$

Les règles suivantes permettent de calculer rapidement les symboles de Jacobi et de Legendre:

$$\left(\frac{-1}{n}\right) = (-1)^{\frac{n-1}{2}} \quad ; \quad \left(\frac{2}{n}\right) = (-1)^{\frac{n^2-1}{8}},$$

multiplicativité $$\left(\frac{ab}{n}\right) = \left(\frac{a}{n}\right)\left(\frac{b}{n}\right)$$

périodicité $$\left(\frac{a+\lambda n}{n}\right) = \left(\frac{a}{n}\right), \qquad \lambda \in \mathbb{Z}$$

loi de réciprocité quadratique: si m et n sont impairs positifs,

$$\left(\frac{n}{m}\right) = (-1)^{\frac{m-1}{2}\frac{n-1}{2}}\left(\frac{m}{n}\right).$$

(f) *Racine carrée modulaire*[3]. Soit p un nombre premier impair et a un carré modulo p. On sait résoudre très rapidement la congruence $x^2 \equiv a \pmod p$. Si $p \equiv 3 \pmod 4$, notons que $x = a^{\frac{p+1}{4}} \bmod p$ est une solution. Si $p \equiv 1 \pmod 4$, on recherche un non carré modulo p, soit b. Un tel b s'obtient par un algorithme probabiliste: on choisit b au hasard, et on calcule $\left(\frac{b}{p}\right)$ jusqu'à ce que $\left(\frac{b}{p}\right) = -1$. Notons que cet algorithme est très efficace, mais que l'on ne connaît pas d'algorithmes déterministes efficaces pour trouver un non carré b lorsque p est grand.

[3]La présentation faite ici est due à Adleman, Manders et Miller (1977). Une forme légèrement différente a été donnée par Tonelli et Shanks.

Hendrik Lenstra, 1988

Supposons a un carré modulo p et

$$a^{e_1} b^{e_2} \equiv 1 \pmod p. \tag{1}$$

La multiplicativité du symbole de Legendre entraîne que e_2 est pair. Une telle relation avec $e_1 = (p-1)/2$ et $e_2 = 0$ est vraie. Si e_1 est impair, alors $x = a^{\frac{e_1+1}{2}} b^{\frac{e_2}{2}}$ est une solution de notre congruence. Si e_1 est pair,

$$u = a^{e_1/2} b^{e_2/2} \equiv \pm 1 \pmod p.$$

Si $u \equiv 1 \pmod p$, on obtient une nouvelle relation (1). Si $u \equiv -1 \pmod p$, alors

$$a^{e_1/2} b^{(e_2+p-1)/2} \equiv +1 \pmod p.$$

Dans les deux cas, on obtient une nouvelle relation (1) avec e_1 remplacé par $e_1/2$. Ceci fournit un algorithme de calcul de la racine carrée modulo p.

Le développement de Hensel permet d'étendre l'algorithme ci-dessus au calcul des racines carrées modulo p^α, et le théorème chinois le généralise modulo n lorsque n est composé et que l'on connaît ses facteurs premiers.

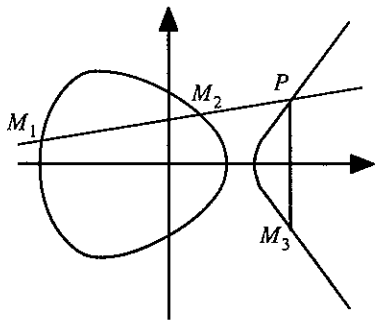
(g) *L'algorithme LLL* (cf. [13], [6]). Introduit par les frères Lenstra et Lovász (d'où son nom) en 1982 pour factoriser les polynômes dans $\mathbb{Z}[x]$, cet algorithme a été utilisé à plusieurs reprises dans d'autres domaines, notamment en cryptographie (cf. 5(b) ci-dessous).

Soit n un entier ≥ 2 et $e = (e_1, e_2, \ldots, e_n)$ une base de $\mathbb{R}^n$. L'ensemble $e_1\mathbb{Z} + e_2\mathbb{Z} + \cdots + e_n\mathbb{Z}$ est appelé réseau de base e. Soit M une matrice à n lignes et n colonnes,

à coefficients entiers et de déterminant ± 1. Soit $e' = (e'_1, e'_2, \ldots, e'_n)$ une nouvelle base de $\mathbb{R}^n$ telle que M soit la matrice de changement de base, alors e' est aussi une base du réseau. Par exemple, le réseau des points à coordonnées entières de $\mathbb{R}^2$ a pour base (1, 0) et (0, 1), mais aussi (100, 3) et (33, 1). Un réseau étant défini par une base, on veut chercher le vecteur non nul le plus court possible dans le réseau, et de façon plus générale une base aussi simple que possible. L'algorithme LLL ne résout pas ces problèmes de façon optimale, mais en fournit en peu de temps une solution raisonnable satisfaisante pour les applications.

3 Les courbes elliptiques

La courbe elliptique $\mathcal{E}_{a,b}$ de paramètres a et b a pour équation $y^2 = x^3 + ax + b$. Elle est symétrique par rapport à l'axe des x. On suppose que le discriminant $\Delta = 4a^3 + 27b^2$ n'est pas nul. Étant donné deux points M_1 et M_2 sur cette courbe, la droite M_1M_2 recoupe la courbe en un troisième point P. On appelle M_3 le symétrique de P par rapport à l'axe des abscisses. Il a été démontré par Jacobi que la loi $(M_1, M_2) \longmapsto M_3$ est une loi de groupe abélien sur l'ensemble des points de $\mathcal{E}_{a,b}$. L'élément neutre est le point à l'infini dans la direction Oy; l'opposé d'un point est son symétrique par rapport à Ox. L'associativité de cette loi n'est pas évidente à démontrer. Cela peut se faire de façon formelle par le calcul en utilisant les formules qui vont suivre (mais le calcul n'est pas simple).



Pour $i = 1, 2, 3$ appelons x_i et y_i les coordonnées de M_i. Il est facile de calculer x_3 et y_3 en fonctions des coordonnées de M_1 et M_2: supposons d'abord $x_1 \neq x_2$. Désignons par $y = \lambda x + \mu$ l'équation de la droite M_1M_2. On a évidemment

$$\lambda = \frac{y_2 - y_1}{x_2 - x_1} \qquad \mu = y_1 - \lambda x_1.$$

Les trois racines de l'équation

$$(\lambda x + \mu)^2 = x^3 + ax + b$$

sont x_1, x_2 et x_3 et leur somme est λ^2. On a donc

$$x_3 = \lambda^2 - x_1 - x_2 \qquad y_3 = -(\lambda x_3 + \mu). \tag{2}$$

Si $x_1 = x_2$, deux cas sont à considérer: si $y_1 + y_2 = 0$, alors M_3 est l'élément neutre du groupe tandis que, si $y_1 = y_2$ on a $M_1 = M_2$, la droite M_1M_2 devient la tangente à la courbe en M_1, sa pente λ se calcule par le théorème des fonctions implicites:

$$\lambda = \frac{dy}{dx} = \frac{3x_1^2 + a}{2y_1}$$

et l'on calcule x_3 et y_3 par (2). Il est commode d'utiliser les coordonnées projectives pour représenter les points: un point à distance finie devient $(x_1 : y_1 : 1)$ et l'élément neutre du groupe s'écrit $(0 : 1 : 0)$.

Tout ce qui a été dit jusqu'à présent peut se faire dans n'importe quel corps. Soit maintenant p un nombre premier différent de 2 et de 3, et $\mathbb{F}_p$ le corps à p éléments. On désignera par $\mathcal{E}_{a,b}(p)$ la courbe elliptique sur $\mathbb{F}_p$ et par $E_{a,b}(p)$ son nombre de points. Hasse a démontré que

$$p + 1 - 2\sqrt{p} \leq E_{a,b}(p) \leq p + 1 + 2\sqrt{p}. \tag{3}$$

De plus, on sait que pour tout t tel que $-2\sqrt{p} < t < 2\sqrt{p}$ il existe au moins une valeur de a et b tel que

$$E_{a,b}(p) = p + 1 - t.$$

Par la loi définie plus haut, les $E_{a,b}(p)$ points forment un groupe fini, et les formules précédentes permettent d'effectuer les opérations dans ce groupe: il faut d'abord calculer λ, c'est-à-dire calculer un quotient modulo p, ce qui se fait, comme nous l'avons vu, par l'algorithme d'Euclide étendu. On calcule ensuite μ et on termine en évaluant les formules (2). Enfin, on peut calculer kP, la somme de k points tous égaux à P, par l'algorithme des puissances.

4 Les tests de primalité

Soit N un nombre entier impair. Un test de primalité doit dire si N est premier ou composé. La méthode des divisions successives consiste à diviser N par tous les nombres premiers au plus égaux à $\sqrt{N}$. Si une division tombe juste, N est composé, et la méthode fournit un diviseur explicite de N. Si aucune division ne tombe juste, N est premier.

Supposons que N ait 31 chiffres décimaux. Le nombre de divisions à effectuer pour s'assurer que N est premier, est supérieur au nombre $\pi(10^{15})$, le nombre de nombres premiers $\leq 10^{15}$, qui vaut 29844570422669 (cf. 8(c) et [22]). Avec un ordinateur effectuant un milliard de divisions à la seconde, cela nécessite 29844 secondes, soit plus de 8 heures, et c'est donc la limite de la méthode.

Comment faire si N a beaucoup plus de 30 chiffres? Le petit théorème de Fermat nous dit que si N est premier et a entier, alors $a^N \equiv a \pmod N$. Si $a^N \not\equiv a \pmod N$, alors N est certainement composé.

Le théorème de Wilson fournit une condition nécessaire et suffisante pour qu'un nombre N soit premier:

$$(N-1)! \equiv -1 \pmod N \iff N \textit{ premier}.$$

Mais il est inutilisable, faute d'un bon algorithme de calcul de $(N-1)!$ mod N. On voit donc que l'algorithmique hiérarchise les théorèmes mathématiques: on peut utiliser le théorème de Fermat, bien qu'il ne soit pas une condition nécessaire et suffisante, car on sait calculer a^{N-1} mod N par l'algorithme des puissances.

Une réciproque du théorème de Fermat est fournie par le théorème de Lucas: on suppose qu'il existe a tel que $a^{N-1} \equiv 1 \pmod N$ et que pour tout q premier divisant $N-1$, $a^{(N-1)/q} \not\equiv 1 \pmod N$; alors N est premier. Notons que, pour appliquer le théorème de Lucas, il faut connaître les facteurs premiers de $N-1$, ce qui est difficile lorsque N est grand. Notons aussi que le théorème de Lucas permet de construire des nombres premiers très grands: soit M un nombre dont tous les facteurs premiers sont connus; pour $x = 1, 2, 3, \ldots$, on regarde si $xM+1$ est premier, et on réussit assez rapidement.

Le théorème de Pocklington (1914) est une amélioration du théorème de Lucas: soit s un diviseur de $N-1$. On suppose qu'il existe a tel que $a^{N-1} \equiv 1 \pmod s$ et que pour tout q premier divisant s, $\operatorname{pgcd}(a^{(N-1)/q}-1, N) = 1$. S'il existe un diviseur premier p de N, alors $p \equiv 1 \pmod s$. De plus, si $s \geq \sqrt{N}$, alors N est premier.

(a) *Les nombres pseudo-premiers.* Différentes sortes de nombres pseudo-premiers ont été introduites pour désigner les nombres non premiers satisfaisant au théorème de Fermat. On dit que n est pseudo-premier en base a (pp-a) si n est composé et si $a^{n-1} \equiv 1 \pmod n$. Exemple: $a = 2$ et $n = 341 = 11 \times 31$. On dit que n est pseudo-premier absolu, ou est un nombre de Carmichael si n est composé et si pour tout a premier avec n, on a $a^{n-1} \equiv 1 \pmod n$. Exemple: $n = 561 = 3 \times 11 \times 17$. Alford, Granville et Pomerance ont récemment prouvé qu'il existait une infinité de tels nombres (cf. [3]).

On dit qu'un nombre impair n est pseudo-premier d'Euler en base a (ppE-a) s'il est composé et si $a^{\frac{n-1}{2}} \equiv \left(\frac{a}{n}\right) \pmod n$, où $\left(\frac{a}{n}\right)$ est le symbole de Jacobi. Enfin la notion de nombres pseudo-premiers forts en base a (ppf-a) est basée sur l'identité:

$$y^{2^s} - 1 = (y-1)(y+1)(y^2+1)\cdots(y^{2^{s-1}}+1). \tag{4}$$

Si n est impair, on écrit $n-1 = 2^s t$ avec $s \geq 1$ et t impair. On dit que l'entier n passe le test ppf-a si, ou bien $a^t \equiv 1 \pmod n$, ou bien, il existe j, $0 \leq j \leq s-1$, tel que $a^{2^j t} \equiv -1 \pmod n$. Notons que, si n est premier et ne divise pas a, par le théorème de Fermat, il divise $a^{2^s t} - 1$ et par (4), il passe le test ppf-a. On dit que n est ppf-a si n est composé, et si n passe le test ppf-a. On sait montrer

$$\text{ppf} - a \Longrightarrow \text{ppE} - a \Longrightarrow \text{pp} - a.$$

Il n'existe pas de nombres pseudo-premiers forts absolus: si n est impair composé, le nombre de a, $1 \leq a \leq n$ tels que n soit ppf-a est inférieur à $n/4$. On en déduit le test de primalité de Miller-Rabin: on choisit k valeurs de a au hasard entre 1 et n, et l'on vérifie que n passe le test ppf-a pour ces k valeurs. Malheureusement ce test ne garantit pas que le nombre n est premier; cependant les exceptions sont très rares.

(b) *Les suites de Lucas.* Le petit théorème de Fermat donne une propriété arithmétique de la suite $u_n = a^n$ qui vérifie $u_n = a u_{n-1}$, et est donc une suite récurrente linéaire d'ordre 1. Une propriété similaire existe pour les suites de Lucas qui sont des suites récurrentes linéaires d'ordre 2. Soit deux entiers P et Q. On désigne par α et β les 2 racines de l'équation $T^2 - PT + Q = 0$. Les suites de Lucas associées aux paramètres P et Q sont définies par

$$U_n = \frac{\alpha^n - \beta^n}{\alpha - \beta} \qquad V_n = \alpha^n + \beta^n$$

ou encore par $U_0 = 0$, $U_1 = 1$, $V_0 = 2$, $V_1 = P$ et

$$U_{n+2} = PU_{n+1} - QU_n \qquad V_{n+2} = PV_{n+1} - QV_n.$$

À l'aide de la relation matricielle

$$\begin{pmatrix} U_{n+1} & V_{n+1} \\ U_n & V_n \end{pmatrix} = \begin{pmatrix} P & -Q \\ 1 & 0 \end{pmatrix}^n \begin{pmatrix} U_1 & V_1 \\ U_0 & V_0 \end{pmatrix}$$

et de l'algorithme des puissances, on peut calculer rapidement U_n et V_n mod$N$. On pose $\Delta = P^2 - 4Q$ et $\varepsilon(N) = \left(\frac{\Delta}{N}\right)$ le symbole de Jacobi, pour N impair. Alors, pour tout p premier impair ne divisant pas $Q\Delta$, $U_{p-\varepsilon(p)}$ est multiple de p.

On peut utiliser cette propriété comme test de primalité: il faut choisir P et Q. Soit N un nombre impair. On peut prendre pour Δ le plus petit nombre de la suite 5, 9, 13, 17, ... tel que le symbole de Jacobi $\left(\frac{\Delta}{N}\right) = -1$. On choisit pour P le plus petit nombre impair plus grand que $\sqrt{\Delta}$ et $Q = (P^2 - \Delta)/4$. Le test de Lucas consiste à vérifier si U_{N+1} est multiple de N.

On appelle nombre pseudo-premier de Lucas (ppL) un nombre composé qui passe ce test. On ne connaît pas à l'heure actuelle de nombres à la fois ppf-2 et ppL (cf. [20]). Pour cette raison, certains systèmes de calcul formel utilisent comme test de primalité un test de Miller-Rabin suivi d'un test de Lucas.

(c) *Le test "sommes de Jacobi"* (cf. [6], voir aussi [16]). Découvert en 1983 par Adleman, Rumely et Pomerance (cf. [1]), ce test a été modifié peu après par Cohen et H. Lenstra qui ont obtenu une version pratique qui, implémentée par Cohen, A. Lenstra et Winter, a permis de tester des nombres de quelques centaines de chiffres en un temps raisonnable.

Curieusement, il aura fallu attendre les années 70 pour que l'on observe que le test d'Euler, $a^{\frac{N-1}{2}} \equiv \left(\frac{a}{N}\right) \pmod N$, ou mieux encore le test pseudo-premier fort étaient plus efficaces que le test de Fermat. L'idée nouvelle ici va être de faire jouer aux petits nombres premiers impairs 3, 5, 7, ... le rôle de 2 dans $a^{\frac{N-1}{2}}$, et pour cela travailler dans le corps des complexes.

En pratique, on va obtenir de nouvelles conditions nécessaires de primalité généralisant le théorème de Fermat et qui s'expriment à l'aide des sommes de Gauss et de

Henri Cohen, 1998

Jacobi. La nouveauté est que si l'on a suffisamment de conditions nécessaires de ce type vérifiées, alors on peut prouver que n est premier. Ce test et sa démonstration ne sont pas simples, et on ne peut les détailler ici, mais il s'agit d'une très belle page de théorie algébrique des nombres. Le temps d'exécution est $O((\log N)^{c \log\log\log N})$, pour une constante c effective.

(d) *Le test ECPP*[4] (cf. [4], [6]). Soit N un nombre non premier. Lorsque $4a^3+27b^2 \not\equiv 0 \pmod N$, on peut définir la courbe $\mathcal{E}_{a,b}(N)$ comme l'ensemble des couples (x, y) vérifiant $y^2 \equiv x^3 + ax + b \pmod N$ auxquels on ajoute le point à l'infini dans la direction Oy. En utilisant les formules (2), on peut faire des opérations sur les points de cette courbe. La seule difficulté est la division par $x_2 - x_1$ lorsque $x_2 - x_1$ n'est pas premier avec N. Mais si cela arrive, alors on est sûr que N n'est pas premier.

Pour prouver la primalité de N, l'idée est de construire une courbe elliptique modulo N dont le nombre de points M se factorise sous la forme

$$M = p_1 p_2 \cdots p_k N_2$$

où $p_1, \ldots, p_k$ sont de petits nombres premiers et N_2 un nombre probablement premier (c'est-à-dire un nombre déclaré premier par un test de Miller-Rabin); puis de démontrer, par un théorème voisin de celui de Pocklington: N_2 premier $\Longrightarrow N$ premier. En recommençant avec N_2, on construit une chaîne décroissante de nombres probablement premiers $N = N_1, N_2, \ldots, N_I$ avec la propriété N_i premier $\Longrightarrow N_{i-1}$ premier. On choisit N_I assez petit pour que l'on puisse prouver simplement sa primalité.

[4]Elliptic Curve Primality Proving.

René J. Schoof, 1988

En 1986, Goldwasser et Killian ont donné une première version théorique de cet algorithme basée sur l'algorithme donné par Schoof pour déterminer le nombre $E_{a,b}(p)$ de points d'une courbe elliptique modulo p. L'algorithme de Schoof est théoriquement rapide, mais inutilisable en pratique. Ensuite Atkin a remplacé l'utilisation de cet algorithme par la construction de courbes elliptiques particulières dont on peut calculer plus simplement le nombre de points. Comme pour le test précédent, il n'est pas facile de décrire complètement ce test ni d'en donner la preuve. Ce test est considéré comme le plus rapide pour de très grands nombres sans forme particulière. Le record actuel est dû à Morain qui en 1995 a montré la primalité du nombre de partitions[5] de 1840926, un nombre de 1505 chiffres décimaux[6].

Le temps d'exécution n'a pas été estimé, mais on conjecture qu'il est en moyenne polynomial en $\log N$ (cf. [6]). De plus la méthode fournit un certificat de primalité (constitué de la suite des nombres N_i, des coefficients a et b des courbes elliptiques utilisées, de leurs nombres de points et d'un point P sur chacune d'elles) qui permet de vérifier assez rapidement que le nombre N est bien premier.

(e) *Les nombres de Mersenne.* Ce sont les nombres premiers de la forme $2^p - 1$, où p est premier. On ignore s'il en existe une infinité. On en connaît 38 (cf. [22]). Les quatre derniers ont été découverts en 1996, 1997, 1998 et 1999 par J. Armengaud ($p = 1398269$) et par G. Spencer ($p = 2976221$), R. Clarkson ($p = 3021377$) et

[5]le nombre de partitions de n est le nombre de façons décrire n comme somme d'entiers positifs: par exemple $n = 5$ a 7 partitions: $5, 4+1, 3+2, 3+1+1, 2+2+1, 2+1+1+1, 1+1+1+1+1$.

[6]En octobre 1997, le record a été battu par E. Mayer et F. Morain avec $\frac{2^{7331}-1}{458072843161}$ qui a 2196 chiffres.

N. Hajratwala ($p = 6972593$) à l'aide d'un programme écrit par G. Woltman et distribué sur PC à 2000 amateurs [7].

Le programme utilise le test de Lucas-Lehmer (qui ne marche que pour les nombres de la forme $2^p - 1$): on construit la suite $u_0 = 4$, $u_{k+1} = u_k^2 - 2 \bmod 2^p - 1$. Pour que $2^p - 1$ soit de Mersenne, il faut et il suffit que $u_{p-2} = 0$. En 1876, Lucas a utilisé ce test pour montrer que $2^{127} - 1$ était premier: pour effectuer les calculs en base 2, il utilisait un échiquier 127×127.

On notera que $2^{6972593} - 1$, qui a 2098960 chiffres, est le plus grand nombre premier connu en décembre 1999.

5 Quelques protocoles de cryptographie

Les méthodes traditionnelles de cryptographie étaient basées sur des permutations de lettres ou de blocs de lettres. Elles ne résistent pas aux techniques linguistiques modernes. Dans le courant des années 70 sont apparues plusieurs méthodes utilisant l'arithmétique. On en trouvera un panorama dans les livres ([23], [24]). Par ailleurs, la cryptographie qui n'intéressait que les militaires et quelques amateurs, a vu son audience s'élargir aux banques, aux entreprises voulant garantir leurs secrets, à la sécurité informatique, etc. Chaque utilisateur recherche le protocole le plus sûr, avec transmission rapide des données. Voici quelques-uns des protocoles parmi les plus utilisés.

(a) *La méthode RSA.* Ce protocole publié en 1978 (cf. [2] ou [24], p. 466) est basé sur le fait qu'on sait construire assez facilement de grands nombres premiers, mais qu'on ne sait pas actuellement trouver les facteurs premiers d'un nombre de plus de 200 chiffres.

Pour construire son code, le chef de réseau

i. construit 2 nombres premiers p et q d'une centaine de chiffres décimaux.

ii. calcule $n = p \times q$

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

iv. choisit e impair au hasard entre 1 et n et tel que $\operatorname{pgcd}(e, \varphi(n)) = 1$ (ceci s'obtient par un algorithme probabiliste: on génère des valeurs de e jusqu'à obtenir satisfaction)

v. calcule d tel que $ed \equiv 1 \pmod{\varphi(n)}$ (ceci se fait en appliquant l'algorithme d'Euclide étendu à e et $\varphi(n)$)

vi. publie dans un annuaire les valeurs de e et n mais garde secrètes les valeurs de $p, q, \varphi(n)$ et d.

Pour envoyer un message au chef de réseau, n'importe qui lit n et e dans l'annuaire. Il faut ensuite transformer le message clair en un (ou plusieurs nombres) M vérifiant

[7] Consulter http://www.mersenne.org/prime.htm

$1 \leq M < n$. Ceci se fait par une méthode simple et décrite dans l'annuaire. Par exemple, si le message est littéral, on remplace la lettre A par 01, la lettre B par 02, la lettre Z par 26, et l'on juxtapose ces différents nombres; si le message contient d'autres caractères, on remplace chaque symbole par son code ASCII [8]. On calcule ensuite par l'algorithme des puissances le cryptogramme ou message chiffré

$$C = M^e \bmod n$$

qui est envoyé au chef de réseau. Celui-ci calcule alors

$$M' = C^d \bmod n$$

à l'aide de son exposant secret de déchiffrage d. Par application du théorème d'Euler ($a^{\varphi(n)} \equiv 1 \pmod{n}$ si a et n sont premiers entre eux), on peut montrer que $M' = M$.

Si un ennemi connaît l'une quelconque des valeurs de p, q, $\varphi(n)$, ou d, on peut montrer qu'il peut en déduire les 3 autres et donc décrypter le message. La fonction qui à p et q fait correspondre $n = pq$ est une fonction "à sens unique": il est facile, à partir de p et q, de calculer n; il est pour l'instant impossible, si p et q ont plus de 100 chiffres, de factoriser n.

Un ennemi peut facilement à partir d'une liste de messages clairs $M_1, M_2, \ldots, M_r$ calculer leurs cryptogrammes correspondants $C_1, C_2, \ldots, C_r$. Est-ce que cette information peut lui permettre de factoriser plus rapidement n? Voilà un type de problèmes mathématiques complètement nouveau soulevé par cette procédure.

Par le protocole RSA que l'on vient de décrire, le chef de réseau ne peut pas distinguer son correspondant habituel d'un ennemi qui voudrait se faire passer pour lui. On peut compléter ce protocole de façon que chaque message soit à la fois secret et signé (cf. [23], [24]).

(b) *La méthode du sac à dos* [9]. Ce protocole est dû à Merkle et Hellman (cf. [24], p. 462). Soit des nombres entiers $c_1, c_2, \ldots, c_k$ vérifiant $0 < c_1 < c_2 < \cdots < c_k$ et un un autre nombre entier C. On veut résoudre l'équation

$$c_1 x_1 + c_2 x_2 + \cdots + c_k x_k = C \qquad x_i \in \{0, 1\}. \tag{5}$$

Dans le sac à dos de volume C, on veut mettre une partie des objets de volume c_1, c_2, etc. de façon à le remplir complètement; $x_i = 1$ veut dire que l'on met l'objet de volume c_i dans le sac, $x_i = 0$, que l'on ne le met pas. Si les nombres c_i sont au hasard, les solutions de (5) sont difficiles à trouver. Il faut regarder les 2^k valeurs possibles des x_i ce qui est infaisable si k est grand. Par contre, si les c_i sont les

[8] American Standard Code for Information Interchange. Par ce code, tous les symboles d'une machine à écrire sont représentés sur un octet, c'est-à-dire qu'on leur associe un nombre entre 0 et 127. Par exemple les lettres majuscules sont représentées par les nombres de 65 (A) à 90 (Z), tandis que les minuscules le sont par les nombres de 97 (a) à 122 (z).

[9] En anglais *knapsack*.

puissances successives de 2, les x_i sont les chiffres de C en base 2, et donc très simples à calculer. On introduit alors la notion de sac à dos facile: les paramètres $c_1, c_2, \ldots, c_k$ doivent vérifier pour tout j tel que $2 \le j \le k$

$$c_1 + c_2 + \cdots + c_{j-1} < c_j.$$

Un sac à dos facile se résout facilement par l'algorithme glouton: dans (5), si $C < c_k$, on doit prendre $x_k = 0$, tandis que, si $C \ge c_k$, on doit choisir $x_k = 1$. Par récurrence descendante, on détermine de même les autres inconnues x_i.

Le protocole cryptographique du sac à dos est alors le suivant:

i. le chef de réseau construit un sac à dos facile $c_1, c_2, \ldots, c_k$

ii. il choisit $N > c_1 + c_2 + \cdots + c_k$

iii. il choisit D premier avec N et calcule D' tel que $DD' \equiv 1 \pmod{N}$ (par l'algorithme d'Euclide étendu)

iv. il calcule $a_i = c_i D \bmod N$

v. il publie $a_1, a_2, \ldots, a_k$ et N dans un annuaire.

Pour envoyer un message au chef de réseau, un correspondant met d'abord son message M sous forme d'une suite de k bits: $m_1, m_2, \ldots, m_k$, avec $m_i \in \{0, 1\}$. Il calcule alors le cryptogramme

$$C = \sum_{i=1}^{k} a_i m_i \bmod N$$

qui est envoyé au chef de réseau.

Celui-ci calcule alors $M' = D'C \bmod N$; un calcul simple dans $\mathbb{Z}/N\mathbb{Z}$ montre que $M' = \sum_{i=1}^{k} c_i m_i$, et comme le sac à dos des c_i est facile, de la valeur de M' il déduit les m_i.

Malheureusement, cette méthode très simple a été cassée par Shamir et Zippel en utilisant l'algorithme LLL (cf. 2(g)).

(c) *Utilisation du logarithme discret.* Rappelons d'abord que le groupe $(\mathbb{Z}/p\mathbb{Z})^*$ est cyclique, pour tout p premier. Nous désignerons par g un générateur. Tout x, $1 \le x \le p-1$ peut donc s'écrire sous la forme $g^a \bmod p$, avec $0 \le a \le p-2$. On appelle a le logarithme discret de x. Comme la fonction $(p, q) \mapsto p \times q$ dans le protocole RSA, la fonction $a \mapsto x = g^a \bmod p$ est une fonction "à sens unique": elle est facile à calculer, mais sa fonction réciproque, le logarithme discret de x, est très difficile à évaluer si p est grand.

i. *Construction d'une clé secrète commune* (Diffie et Hellman, cf. [24], p. 519). Alice et Bob veulent calculer un secret commun. Ils conviennent d'un nombre premier p de 200 chiffres, et d'un générateur g; p et g sont publics.

Alice choisit a au hasard, $1 \le a \le p-1$ et calcule $\alpha = g^a \bmod p$. Elle publie α dans un annuaire, et garde a secret.

Bob fait de même avec b et $\beta = g^b \bmod p$.

Alice lit β dans l'annuaire, et calcule $k = \beta^a \bmod p$. Bob lit α dans l'annuaire, et calcule $k' = \alpha^b \bmod p$. Il est facile de voir que $k = k'$, et ainsi Alice et Bob ont une clé commune qu'ils peuvent utiliser dans un protocole de cryptographie à clé secrète.

ii. *Échange de messages* (Shamir, cf. [24] p. 517). Comme précédemment, Alice et Bob conviennent d'un nombre premier p et d'un générateur g publics. Alice choisit a, $1 \le a \le p-1$ et premier avec $p-1$. Elle calcule a' tel que $aa' \equiv 1 \pmod{p-1}$. Bob fait de même avec b et b'; a, a', b et b' restent secrets. Le message est mis sous forme d'un (ou plusieurs) nombre M tel que $1 \le M \le p-1$. Pour envoyer un message, Alice et Bob procèdent en 4 temps:

A. Alice envoie à Bob $C_1 = M^a \bmod p$.

B. Bob calcule $C_2 = C_1^b \bmod p$ et l'envoie à Alice.

C. Alice calcule $C_3 = C_2^{a'} \bmod p$ et l'envoie à Bob.

D. Bob calcule $C_4 = C_3^{b'} \bmod p$. Par le théorème de Fermat, on voit que $C_4 = M$.

Ce protocole s'appelle protocole des valises: dans le premier temps, Alice met un cadenas a sur la valise qu'elle envoie à Bob. Dans le deuxième temps, Bob met un autre cadenas b sur la valise. Dans le troisième temps, Alice enlève son cadenas. À la fin, il ne reste plus à Bob qu'à enlever son propre cadenas pour pouvoir ouvrir la valise.

Si un ennemi sait calculer le logarithme discret modulo p, il saura aisément décrypter le message M. Mais est-il possible de décrypter le message par une autre méthode? Voilà un problème mathématique de nature très différente des problèmes classiques, et que l'on ne sait guère comment aborder.

(d) *Jouer à pile ou face par téléphone.* Il existe plusieurs protocoles pour jouer à pile ou face sans que les joueurs s'observent, et bien sûr sans tricher (cf. [24]). Celui que nous allons décrire est basé sur les racines carrées modulo n.

Soit p et q deux nombres premiers impairs, et $n = p \times q$. Si a est premier avec n, la congruence $x^2 \equiv a \pmod{n}$ a 4 solutions si les deux symboles de Legendre $\left(\frac{a}{p}\right)$ et $\left(\frac{a}{q}\right)$ valent $+1$ et n'en a pas sinon. Par exemple, la congruence $x^2 \equiv 4 \pmod{15}$ admet comme solutions 2, 7, 8 et 13, c'est-à-dire ± 2 et ± 7. Si l'on connaît p et q, même s'ils sont très grands, la congruence $x^2 \equiv a \pmod{n}$ est rapide à résoudre: on résout $x^2 \equiv a$ modulo p puis modulo q par la méthode exposée en 2 (e) et l'on recompose les solutions obtenues par le théorème chinois (cf. 2(c)). Réciproquement, si l'on connaît les 4 racines carrées de a modulo n, soit $\pm\gamma$ et $\pm\delta$ avec $1 \le \gamma < \delta \le (n-1)/2$, alors $\operatorname{pgcd}(\delta - \gamma, n)$ et $\operatorname{pgcd}(\delta + \gamma, n)$ valent p et q.

Autrement dit, la connaissance des 4 racines carrées de a modulo n est équivalente à la connaissance des facteurs premiers de n.

Et maintenant, voici le protocole:

i. Alice construit deux nombres premiers p et q d'une centaine de chiffres et calcule $n = pq$. Elle envoie n a Bob. Comme p et q sont grands, Bob ne peut pas factoriser n.

ii. Bob vérifie que n n'est pas une puissance de nombre premier et qu'il est impair (pour s'assurer qu'Alice n'a pas triché). Il choisit x au hasard entre 2 et $n-2$ et envoie à Alice $a = x^2 \bmod n$.

iii. Alice (qui connaît p et q) calcule les 4 racines carrées de a mod n, par la méthode exposée ci-dessus, et en envoie une au hasard, soit y, à Bob.

iv. Si $y \neq \pm x$, Bob gagne: il connaît les 4 racines carrées de a, il calcule p et q, et les envoie à Alice pour montrer qu'il a gagné.

Si $y = \pm x$, Bob a perdu: la connaissance de y ne lui apprend rien de nouveau. Il avoue sa défaite à Alice, qui lui fournit les valeurs de p et q pour montrer qu'elle n'a pas triché.

Notons que, si Alice choisissait pour n un produit de 3 nombres premiers (ou plus), elle se désavantagerait. Le protocole suppose donc que les joueurs veulent gagner. On ne sait pas à l'heure actuelle "casser" ce protocole. Par contre dans un protocole assez voisin de "poker mental" (pour jouer aux cartes par téléphone), une possibilité assez subtile de s'avantager pour l'un des joueurs a été trouvée (cf. [24], p. 94).

(e) *Le protocole de Feige, Fiat et Shamir (1986) ou comment persuader quelqu'un que l'on connaît un secret sans en rien révéler* (cf. [24], p. 503). Alice est cliente de la banque et Bob est le banquier. La banque construit deux nombres premiers p et q et calcule $n = p \times q$ comme dans RSA. Personne ensuite n'a plus besoin de connaître p et q. Comme chaque client, Alice a un secret $s \in \{1, 2, \ldots, n-1\}$. Sur sa carte, est inscrit $v = s^2 \bmod n$. Comme nous l'avons vu en (d) déterminer s à partir de v est équivalent à factoriser n, ce qui est impossible si n a 200 chiffres. Le protocole de reconnaissance de Alice par Bob est le suivant:

i. Alice choisit x au hasard entre 2 et $n-1$. Elle calcule $t = x^2 \bmod n$ et donne t à Bob.

ii. Bob donne à Alice $c = 0$ ou $c = 1$ avec probabilité $1/2$.

iii. Alice donne à Bob le nombre $y = xs^c \bmod n$, c'est-à-dire $y = x$ si $c = 0$ et $y = xs \bmod n$ si $c = 1$.

iv. Bob vérifie que $y^2 \equiv v^c t \pmod n$, et si oui, reconnaît Alice avec une probabilité $1/2$.

Notons qu'un escroc $\tilde{A}$ qui veut se faire passer pour Alice, peut le faire avec une chance sur deux de réussir: s'il anticipe $c = 0$, il sait calculer $y = x$ sans utiliser le secret s et s'il anticipe $c = 1$, il peut donner au lieu de t, $\tilde{t} = x^2 v^{-1} \bmod n$ puis, au lieu de y, $\tilde{y} = x$ et Bob vérifiera que $\tilde{y}^2 \equiv v\tilde{t} \pmod n$.

Si l'on exécute 20 fois ce protocole de reconnaissance, la probabilité pour un escroc de se faire passer pour Alice devient 2^{-20}, soit environ une chance sur un million ce qui est très faible. En utilisant la théorie de la zéro connaissance (zero knowledge) Feige, Fiat et Shamir ont démontré que, même exécuté de nombreuses fois, ce protocole ne donne aucune information à Bob sur le secret d'Alice.

6 Méthodes de factorisation

La recherche des facteurs premiers des nombres entiers a toujours intéressé les mathématiciens, mais il faut reconnaître que faire les calculs à la main n'est pas très gratifiant. L'arrivée des ordinateurs a relancé l'intérêt pour ce problème, et surtout ses retombées possibles pour la cryptographie ont augmenté considérablement les efforts de recherche dans ce domaine. Nous décrivons ci-dessous les principales méthodes découvertes pendant les 25 dernières années. On trouvera une description plus complètes de ces méthodes, et de quelques autres dans [9], [10] ou [6]. On constatera la grande variété des outils utilisés, comme souvent en théorie des nombres, pour résoudre un problème dont l'énoncé est vraiment élémentaire. Nous commençons par les propriétés de la fonction Ψ qui nous servira à évaluer le temps d'exécution de certains des algorithmes suivants.

(a) *La fonction $\Psi(x, y)$ de de Bruijn.* Soit $P(n)$ le plus grand facteur premier de n. On dit que n est friable si tous ses facteurs premiers sont petits, et plus précisément, on dit que n est y-friable, si $P(n) \leq y$. La fonction $\Psi(x, y)$ compte le nombre de nombres y-friables entre 1 et x:

$$\Psi(x, y) = \sum_{n \leq x,\ P(n) \leq y} 1.$$

Cette fonction a été étudiée par de Bruijn, Canfield, Erdős et Pomerance, Hildebrand et Tenenbaum (cf. [7], [5], [8]). Nous donnons seulement les résultats nécessaires pour évaluer les algorithmes de factorisation suivants. Soit $u = \frac{\log x}{\log y}$. Pour tout $\varepsilon > 0$, on a lorsque $u \to \infty$ avec $y \geq (\log x)^{1+\varepsilon}$

$$\Psi(x, y) = \frac{x}{u^{u(1+o(1))}}. \tag{6}$$

Soit

$$L(x) = \exp(\sqrt{\log x \log\log x}). \tag{7}$$

En utilisant (6), on peut montrer que pour tous nombres réels $a, b > 0$, on a

$$\Psi(x^a, L(x)^b) = \frac{x^a}{L(x)^{a/(2b)+o(1)}}, \qquad x \to \infty. \tag{8}$$

(b) *La méthode ρ de Pollard* (cf. [19]). Cette méthode se programme en quelques lignes, et est basée sur les probabilités et les ensembles finis. Tout le monde connaît le paradoxe des anniversaires: si l'on réunit n personnes, il faut que $n > 366$ pour être absolument certain que deux de ces personnes auront leur anniversaire le même jour; mais si $n \geq 23$ alors la probabilité de cet événement est supérieur à $1/2$. Ceci se généralise de la façon suivante: la probabilité qu'une application d'un ensemble de n éléments dans un ensemble à m éléments soit injective est

$$p(m,n) = \frac{m(m-1)\cdots(m-n+1)}{m^n} = \prod_{k=1}^{n-1}\left(1-\frac{k}{m}\right) \leq \exp\left(-\frac{n(n-1)}{2m}\right).$$

Pour $m = 365$ et $n = 23$, on a bien $\exp(-\frac{n(n-1)}{2m}) = 0.4999982\ldots < \frac{1}{2}$.

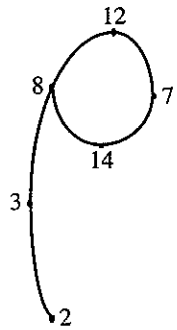
Soit E un ensemble à m éléments, f une application de E dans E et x_0 un point de E. On considère la suite définie par $x_1 = f(x_0)$, $x_2 = f(x_1)$, etc. Tout processus fini est périodique: il existe une queue (éventuellement vide) $x_0, \ldots, x_{\mu-1}$ de points qui ne réapparaissent pas dans la suite, et une période $\lambda \geq 1$ telle que l'on ait pour $k \geq \mu$,

$$x_{k+\lambda} = x_k.$$

On définit encore l'épacte e de la suite comme le plus petit entier k tel que $x_{2k} = x_k$. On peut montrer que

$$\mu \leq e \leq \mu + \lambda - 1.$$

Exemple: $m = 17$, $E = \{0, 1, 2, \ldots, 16\}$, $f(x) = x^2 - 1 \bmod 17$, $x_0 = 2$. Les premiers termes de la suite x_k sont 2, 3, 8, 12, 7, 14, 8, 12, 7, ... de telle sorte que la queue $\{2, 3\}$ a pour longueur $\mu = 2$, la période est $\lambda = 4$ et l'épacte est $e = 4$. Notons que ces points dessinent un ρ d'où le nom de la méthode:



En supposant les m valeurs possibles pour x_0 et les m^m fonctions f possibles comme équiprobables, on peut montrer (et ceci s'apparente au paradoxe des anniversaires) que les valeurs moyennes $E(\lambda)$ et $E(\mu)$ vérifient, lorsque m tend vers l'infini:

$$E(\lambda) = E(\mu) \sim \sqrt{\frac{\pi m}{8}} = 0.62\ldots\sqrt{m}\,;\ E(e) \sim \sqrt{\frac{\pi^5 m}{288}} = 1.03\ldots\sqrt{m}.$$

Soit maintenant p un nombre premier, $E = \{0, 1, \ldots, p-1\}$, c un nombre entier, et $f(x) = x^2 + c \bmod p$. On désigne par $e(p,c)$ l'épacte de la suite $x_0 = 2$, $x_1 = f(x)$, etc. On a calculé par ordinateur, que pour $p < 10^6$, et $c = \pm 1$, on a $e(p,c) \leq 3800$, et l'on conjecture que pour tout p premier, on a

$$e(p, \pm 1) \leq \frac{4}{3}\sqrt{p \log p}. \tag{9}$$

Soit maintenant un nombre N à factoriser (garanti non premier par un test du paragraphe 4). On exécute l'algorithme suivant avec $x0 = 2$ et $c = \pm 1$:

fonction rho(x0, c, N)
X:=x0;
Y:=x0;
pour k:= 1 à kmax *faire*
X:=X*X + c mod N;
Y:=Y*Y + c mod N;
Y:=Y*Y + c mod N;
D:=pgcd(Y-X,N);
si D≠ 1, écrire D, et s'arrêter; *finsi;*
finpour;
fin.

Soit z_k la suite de nombres entiers naturels définis par $z_0 = 2$ et $z_{k+1} = z_k^2 + c$. On remarque que dans la boucle, juste avant le calcul de D, la mémoire X contient $z_k \bmod N$, et Y contient $z_{2k} \bmod N$.

Soit p un facteur premier de N, et soit $e = e(p,c)$ l'épacte de la suite x_k définie précédemment, et qui est exactement $x_k = z_k \bmod p$. Que se passe-t-il lorsque $k = e$ dans l'algorithme ci-dessus? On a $x_k = x_{2k}$ par définition de l'épacte, ce qui se traduit par "p divise $z_{2k} - z_k$". Comme p divise N, p divise $Y - X$ qui vaut $z_{2k} - z_k \bmod N$ et ainsi p divise D qui sera différent de 1. L'algorithme s'arrêtera donc sûrement lorsque $k = e(p,c)$.

Si N n'est pas premier, son plus petit facteur premier p est $\leq \sqrt{N}$, et sous la conjecture ci-dessus, on aura

$$k \leq e(p,c) \leq \frac{4}{3}\sqrt{p \log p} \leq \frac{4}{3\sqrt{2}} N^{1/4}\sqrt{\log N}.$$

On a donc un algorithme qui, sous la conjecture (9), est en $O(N^{1/4+o(1)})$.

(c) *La méthode $p-1$ de Pollard (1975)*, (cf. [18] et [6]). Soit N un nombre garanti non premier et soit p un facteur premier de N. Soit k un nombre tel que $p-1$ divise $k!$. Notons que cette condition est très voisine de la condition "$p-1$ est k-friable". Que vaut $\text{pgcd}(a^{k!}-1, N)$? Par le théorème de Fermat, ce pgcd est multiple de p. Par ailleurs, par l'algorithme des puissances (cf. 2(c)), $a^{k!} \bmod N$ se calcule en $O(\log(k!))$ soit en $O(k \log k)$ opérations. On a donc une méthode qui permet de trouver rapidement les facteurs premiers p de N tels que $p-1$ n'ait que des petits facteurs premiers.

La fonction factorielle n'est pas indispensable: au lieu de $k!$ on peut prendre $\prod_{p \le k,\, p \text{ premier}} p^{a_p}$ où a_p est la partie entière de $\frac{\log k}{\log p}$.

Pour éviter qu'un ennemi puisse factoriser n par cette méthode, on choisit toujours dans le protocole RSA des nombres premiers p et q, tels que $p-1$ et $q-1$ aient un grand facteur premier.

(d) *La méthode des courbes elliptiques de H.W. Lenstra (1985)*, (cf. [11], [6]). La méthode $p-1$ vue en (b) utilise les propriétés du groupe $(\mathbb{Z}/p\mathbb{Z})^*$ en espérant que $p-1$ soit friable. Mais s'il ne l'est pas, on ne peut rien faire, d'où l'idée de trouver une famille de groupes d'ordre voisin de p et dont certains auront un ordre friable. On va utiliser les courbes elliptiques $\mathcal{E}_{a,b}(p)$ qui, par la relation (3) ont un ordre voisin de p. On conjecture que la probabilité que $E_{a,b}(p)$ soit y-friable est proche de $\frac{1}{p}\Psi(p, y)$ qui est la probabilité qu'un nombre au hasard inférieur à p soit y-friable.

On veut factoriser N non premier. On pose $L = L(N)$ où la fonction L est définie par (7) et on choisit $K = L^{\beta}$.

i. *Choix d'une courbe elliptique au hasard.* On choisit au hasard a, x_0, y_0 entre 1 et N et l'on calcule b pour que

$$y_0^2 \equiv x_0^3 + ax_0 + b \pmod{N}.$$

On s'assure que $4a^3 + 27b^2$ et N sont premiers entre eux. Soit p un diviseur premier de N. On travaillera en pensée sur la courbe $\mathcal{E}_{a,b}(p)$, mais comme on ne connaît pas p on effectuera les opérations modulo N.

ii. *Calcul de $k! \cdot P$.* Le point $P = (x_0, y_0)$ est sur la courbe $\mathcal{E}_{a,b}(p)$ par construction. Pour $k \le K$, on calcule $k! \cdot P$ en utilisant l'algorithme des puissances 2(c), et les formules (2) d'addition des points modulo N. Le seul problème est le calcul de λ qui nécessite une division. Si le diviseur est premier avec N, on obtient le quotient en leur appliquant l'algorithme d'Euclide étendu (cf. 2(b)). Si le diviseur n'est pas premier avec N, alors leur pgcd fournira un facteur de N que l'on peut espérer $\neq N$. Si le calcul de $k! \cdot P$ pour $k \le K$ ne s'arrête pas, on recommence avec une autre courbe.

Supposons que $p \le \sqrt{N}$ soit un facteur premier de N. On a donc $p = N^{\alpha}$ avec $\alpha \le 1/2$. Sous la conjecture énoncée précédemment, la probabilité que $E_{a,b}(p)$ soit

friable est par la formule (8) voisine de

$$\frac{1}{p}\Psi(p, K) = \frac{1}{N^{\alpha}}\Psi(N^{\alpha}, L^{\beta}) = \frac{1}{L^{\frac{\alpha}{2\beta}+o(1)}}.$$

Si l'on choisit donc un peu plus que $L^{\frac{\alpha}{2\beta}}$ courbes, on peut espérer en trouver une dont l'ordre soit K-friable, et pour laquelle une division modulo N ne pourra pas se faire, révélant un facteur de N. Comme, pour chaque courbe le calcul de $k! \cdot P$ nécessite $O(K \log K) = L^{\beta+o(1)}$ opérations, l'algorithme requiert $L^{\beta+\frac{\alpha}{2\beta}+o(1)}$ pas. En choisissant $\beta = \sqrt{\alpha/2}$ on obtient $L^{\sqrt{2\alpha}+o(1)}$ pas, et comme $\alpha \le 1/2$, l'algorithme est en $O(L^{1+o(1)})$ pas.

Cet algorithme est actuellement le meilleur connu pour trouver des facteurs premiers pas trop grands (jusqu'à 40 ou 45 chiffres) [10] de nombres N éventuellement grands (jusqu'à 200 chiffres).

(e) *Les méthodes à combinaisons de congruences.* L'idée de ces méthodes remonte à Kraitchik aux environs de 1920.

i. *La base de nombres premiers.* On se donne un ensemble $\mathcal{B} = \{p_1, p_2, \ldots, p_k\}$, où $p_1 = -1$, et $p_2, p_3, \ldots, p_k$ sont les nombres premiers inférieurs à une borne fixée B. Soit N un nombre impair à factoriser. On suppose que l'on connaît des nombres entiers $Q_1, Q_2, \ldots, Q_r$ et $u_1, u_2, \ldots, u_r$, avec $r > k$ tels que

A. les Q_i sont des carrés modulo N:

$$Q_i \equiv u_i^2 \pmod{N} \tag{10}$$

B. les Q_i se factorisent complètement sur la base

$$Q_i = \prod_{j=1}^{k} p_j^{\alpha_{i,j}}. \tag{11}$$

Les différentes méthodes se distingueront par la façon d'obtenir les familles de nombres Q_i et u_i.

La matrice $(\alpha_{i,j})$ a r lignes et k colonnes. On pose $a_{i,j} = \alpha_{i,j} \bmod 2$ de telle sorte que $a_{i,j} \in \mathbb{F}_2 = \mathbb{Z}/2\mathbb{Z}$. Comme $r > k$, les vecteurs lignes v_i de la matrice $a_{i,j}$ sont liés dans $\mathbb{F}_2^k$. Par une variante de la méthode du pivot de Gauss, on trouve une relation de liaison:

$$v_{i_1} + v_{i_2} + \cdots + v_{i_s} = 0, \qquad s \le r. \tag{12}$$

[10] Le record actuel est dû à Curry qui a trouvé en 1998 un facteur premier de 53 chiffres à un facteur de 150 chiffres du nombre $2^{677}-1$ dont la factorisation est maintenant complète.

En posant

$$x = \prod_{1 \le j \le k} p_j^{(\alpha_{i_1,j}+\cdots+\alpha_{i_s,j})/2} \bmod N \qquad \text{et} \qquad y = \prod_{j=1}^{s} u_{i_j} \bmod N,$$

on obtient

$$x^2 \equiv y^2 \equiv Q_{i_1} Q_{i_2} \cdots Q_{i_s} \pmod N. \tag{13}$$

Le calcul des pgcd de $x - y$ et N puis de $x + y$ et N fournira en général des facteurs non triviaux de N.

ii. *La méthode de Dixon.* On prend un nombre B et la base $\mathcal{B}$ est formée des nombres premiers $\le B$. On choisit ensuite très simplement A au hasard entre 1 et N le nombre à factoriser, et l'on calcule $Q(A) = A^2 \bmod N$. Puis, on divise $Q(A)$ par les nombres premiers de la base $\mathcal{B}$ et si $Q(A)$ se factorise complètement sur $\mathcal{B}$, on garde $Q(A)$ et A comme valeur de Q_i et u_i. Cette méthode est intéressante du point de vue théorique, mais elle n'a jamais été utilisée en pratique.

iii. *La méthode CFRAC de Morrison et Brillhart (1975).* On calcule le développement en fractions continues de $\sqrt{N}$. Soit $\frac{p_k}{q_k}$ la k-ème réduite. La théorie des fractions continues nous donne

$$p_k^2 - N q_k^2 = v_k, \qquad |v_k| \le 2\sqrt{N}.$$

On a donc $v_k \equiv p_k^2 \pmod N$. On calcule v_k pour $k = 1, 2$, etc., on regarde si v_k se factorise sur la base de nombres premiers $\mathcal{B}$ choisie et si oui, on garde v_k et p_k comme valeur de Q_i et u_i. L'avantage sur la méthode de Dixon est que, comme v_k est de l'ordre de grandeur de $\sqrt{N}$, il se factorisera beaucoup plus souvent sur la base $\mathcal{B}$ que $Q(A)$ qui était de l'ordre de grandeur de N. Les deux auteurs ont illustré leur méthode en factorisant le nombre de Fermat $F_7 = 2^{2^7} + 1$, qui a 39 chiffres décimaux.

iv. *Le crible quadratique de C. Pomerance (1981)*, (cf. [21]). Considérons le polynôme

$$f(A) = (\lfloor \sqrt{N} \rfloor + A)^2 - N.$$

Pour $A = N^{o(1)}$, on a $f(A) \sim 2A\sqrt{N}$ et donc $f(A)$ ne dépasse guère $\sqrt{N}$. Comme $f(A)$ est un carré modulo N, on pourra garder comme couple (Q_i, u_i) les couples $(f(A), \lfloor \sqrt{N} \rfloor + A)$ lorsque $f(A)$ est B-friable.

Pour repérer les $f(A)$ qui sont B-friables, on utilise une technique de crible. Si $f(A)$ est multiple de p, alors $f(A + p)$, $f(A + 2p)$, etc., sont aussi multiples de p. On fait varier A dans l'intervalle $[-A_{\max}..A_{\max}]$; on initialise un tableau $T(A)$ à 1 puis, pour chaque nombre premier $p \in \mathcal{B}$ on multiplie $T(A)$ par p pour chaque A tel que $f(A)$ est multiple de p. Ceci se réalise en résolvant la congruence $f(A) \equiv 0 \pmod p$. Si p ne divise pas N (ce qui est évidemment contrôlé), cette congruence a 0 ou 2 solutions: A_1 et A_2. Les A concernés sont alors $A_1 + \lambda p$ et $A_2 + \lambda p$ avec $\lambda = 0, \pm 1, \pm 2, \ldots$. Ce que l'on vient de faire avec p, on le refait avec p^2, p^3, etc. À la fin, les A tels que $T(A) = f(A)$ sont certainement B-friables. Et si l'on a $T(A) < f(A)$, c'est que $f(A)$ a un facteur premier supérieur à B donc $T(A) < f(A)/B$ et $T(A)$ est nettement plus petit que $f(A)$.

Le grand intérêt de la méthode du crible quadratique est qu'elle peut se pratiquer non pas sur les nombres entiers $T(A)$ et $f(A)$, mais sur les valeurs approchées de ces nombres en virgule flottante, ce qui se fait beaucoup plus vite.

Pour évaluer le temps d'exécution, on pose $L = L(N)$ où la fonction L est définie par (7). On choisit $B = L^b$ et le nombre k d'éléments dans la base $\mathcal{B}$ sera $L^{b+o(1)}$. On admet que les nombres $f(A)$ se comportent du point de vue de la friabilité comme des nombres au hasard voisin de $\sqrt{N}$, c'est-à-dire que la probabilité que $f(A)$ soit B-friable est proche de

$$\frac{\Psi(N^{1/2}, L^{b+o(1)})}{N^{1/2}} = \frac{1}{L^{\frac{1}{4b}+o(1)}},$$

par (8). Pour trouver un peu plus de k valeurs de A telles que $f(A)$ soit B-friables, il faudra choisir $A_{\max}$ un peu plus grand que k fois $L^{b+o(1)}$, soit $A_{\max} = L^{b+\frac{1}{4b}+o(1)}$. Le coût du criblage est à peine plus élevé et vaut

$$O(A_{\max} \sum_{p \le B} \frac{1}{p}) = O(A_{\max} \log\log B) = L^{b+\frac{1}{4b}+o(1)}.$$

L'algorithme du pivot de Gauss effectué sur une matrice à k lignes et k colonnes est en $O(k^3)$. Mais, pour une matrice à coefficients dans $\mathbb{F}_2$, on peut utiliser l'algorithme de Wiedemann (cf. [27]) qui est en $O(k^{2+o(1)}) = L^{2b+o(1)}$. Au total, le nombre de pas est $O(L^{b+\frac{1}{4b}+o(1)} + L^{2b+o(1)}) = L^{1+o(1)}$ en choisissant $b = 1/2$.

Diverses améliorations ont été apportées pour augmenter l'efficacité de cette méthode, notamment l'utilisation de plusieurs polynômes qui permet de garder $f(A)$ très près de $\sqrt{N}$ et de travailler indépendemment sur plusieurs machines. En 1984, J. Davis et D. Holdridge assuraient le succès du crible quadratique en factorisant le nombre $\frac{10^{71}-1}{9}$ qui s'écrit avec 71 chiffres 1. En 1994, un nombre de 129 chiffres était factorisé, en utilisant la programmation distribuée sur de nombreuses machines.

(f) *Le crible du corps de nombres* (cf. [12], [21]). Lancée en 1988 par John Pollard, cette nouvelle méthode était à l'origine destinée à factoriser des nombres de forme particulière, par exemple $2^k \pm 1$. Elle attira l'attention lorsqu'en 1990, les frères Lenstra et Manasse l'utilisèrent pour factoriser le nombre de Fermat $F_9 = 2^{2^9} + 1$ qui a 155 chiffres.

Pour factoriser N, la première étape consiste à construire un polynôme $f \in \mathbb{Z}[x]$ unitaire, irréductible sur $\mathbb{Q}$, de degré pas trop élevé (5 ou 6 pour un nombre N de 100 à 200 chiffres) et un nombre m tel que $f(m) \equiv 0 \pmod N$. Ceci est facile pour les nombres de la forme $2^k \pm 1$: pour $N = F_9 = 2^{512} + 1$, on peut choisir $f(x) = x^5 + 8$ et $m = 2^{103}$. Pour un nombre N au hasard, on fixe d'abord d, puis $m = \lfloor N^{1/d} \rfloor$. On écrit ensuite N dans la base de numération m

$$N = m^d + c_{d-1}m^{d-1} + \cdots + c_1 m + c_0, \qquad 0 \le c_i \le m-1,$$

et l'on pose $f(x) = x^d + c_{d-1}x^{d-1} + \cdots + c_0$. En général, ce polynôme est irréductible sur $\mathbb{Z}[x]$; s'il ne l'est pas, on peut montrer que l'on en déduit une factorisation non triviale de N.

Soit α une racine complexe de f, et considérons l'anneau $\mathbb{Z}[\alpha] \cong \mathbb{Z}[x]/f(x)\mathbb{Z}[x]$. Comme $f(\alpha) = 0$ et $f(m) \equiv 0 \pmod N$, l'application qui à α fait correspondre m se prolonge en un homomorphisme d'anneau ϕ de $\mathbb{Z}[\alpha]$ dans $\mathbb{Z}/N\mathbb{Z}$.

Supposons maintenant que l'on connaisse un ensemble A de couples (a, b) avec a et b premiers entre eux possédant deux propriétés: le produit des éléments $a - \alpha b$ pour tous les couples (a, b) de A est un carré dans $\mathbb{Z}[\alpha]$, disons γ^2, et le produit des nombres $a - mb$ pour tous les couples (a, b) de A est un carré v^2 dans $\mathbb{Z}$. Posant $u = \phi(\gamma)$, on a

$$\begin{aligned} u^2 = \phi(\gamma)^2 = \phi(\gamma^2) &= \phi\Big(\prod_{(a,b)\in A}(a-\alpha b)\Big) = \prod_{(a,b)\in A}\phi(a-\alpha b) \\ &\equiv \prod_{(a,b)\in A}(a-mb) = v^2 \pmod N \end{aligned}$$

et comme dans (13) ci-dessus, $\operatorname{pgcd}(u - v, N)$ et $\operatorname{pgcd}(u + v, N)$ fourniront des facteurs de N.

Pour trouver un ensemble A, on peut suivre la méthode de la base de nombres premiers exposée ci-dessus en (e)i. Supposons que $\mathbb{Z}[\alpha]$ soit l'anneau des entiers du corps $\mathbb{Q}[\alpha]$ et que de plus il soit principal (ces deux conditions sont loin d'être toujours réalisées). On choisit alors une base $\mathcal{B} \in \mathbb{Z}[\alpha]$ formée d'éléments premiers auxquels on ajoute les unités fondamentales. On choisit une base $\mathcal{B}'$ dans $\mathbb{Z}$ formée de nombres premiers et de -1 comme dans (e)i. Par une méthode de crible, on va sélectionner des couples (a_i, b_i) tels que $a_i - b_i\alpha$ se factorise sur $\mathcal{B}$ et $a_i - b_i m$ sur $\mathcal{B}'$. On considère alors la matrice dont la i-eme ligne est formée des exposants de $a_i - b_i\alpha$ dans $\mathcal{B}$ puis des exposants de $a_i - b_i m$ dans $\mathcal{B}'$. Si le nombre de lignes de cette matrice est supérieur à son nombre de colonnes on en déduit comme dans (e)i la détermination d'un ensemble A.

Pour étendre la méthode au cas général où $\mathbb{Z}[\alpha]$ n'a pas les propriétés supposées ci-dessus, il a fallu un travail important en théorie algébrique des nombres (cf. [12], [21], [6]). Sous des hypothèses raisonnables, et en utilisant l'estimation (6), on peut montrer que la méthode GNFS (General Number Field Sieve) est en $O(L_N(1/3, c+o(1))$, avec

$$L_N(s, c) = \exp(c(\log N)^s(\log\log N)^{1-s}) \tag{14}$$

et $c = (64/9)^{1/3} = 1,923\ldots$. En octobre 1999, un nombre de 155 chiffres sans forme spéciale a été factorisé par GNFS, et il semble qu'à partir de cette taille, GNFS soit définitivement supérieur à la méthode du crible quadratique. La recherche des relations de dépendance entre les lignes d'une matrice à coefficients dans $\mathbb{F}_2$ intervient dans ces deux algorithmes. Dans les débuts du crible quadratique, cette recherche prenait un temps négligeable, mais elle devient de plus en plus longue et difficile avec des matrices dont le nombre de lignes ou de colonnes avoisinent un million.

7 Calcul du logarithme discret

(cf. [17], [26]). Due semble-t-il à Kraitchik, puis reprise par Western et Miller, la première méthode de calcul du logarithme discret s'inspire du calcul des logarithmes népériens: on considère la série

$$\frac{1}{2}\log\left(\frac{1+x}{1-x}\right) = x + \frac{x^3}{3} + \frac{x^5}{5} + \cdots,$$

en choisissant x le plus petit possible pour que la convergence soit rapide. Par exemple, pour calculer $\log 2$ et $\log 3$, on utilise $x = 1/17$ qui fournira $\log\frac{9}{8} = 2\log 3 - 3\log 2$ et $x = \frac{5}{59}$ qui donnera $\log\frac{32}{27} = 5\log 2 - 3\log 3$. Par résolution du système linéaire, on en déduit $\log 2$ et $\log 3$.

On s'inspire également des méthodes de factorisation à combinaison de congruences: soit P un nombre premier et g un générateur de $\mathbb{F}_P^*$. On se fixe une base de nombres premiers $\mathcal{B}$, et $B = \max \mathcal{B}$. Dans un premier temps, on veut calculer $\log_g p$ pour tout $p \in \mathcal{B}$. Pour cela, on calcule $g^k \bmod P$, $k = 1, 2, 3, \ldots$, et l'on ne garde que les k_i pour lesquels $g^{k_i} \bmod P$ se factorise sur la base $\mathcal{B}$:

$$g^{k_i} \bmod P = \prod_{p\in\mathcal{B}} p^{\alpha_{i,p}}, \qquad 1 \le i \le I,$$

qui se traduit par

$$k_i \equiv \sum_{p\in\mathcal{B}} \alpha_{i,p}\log_g p \pmod{p-1}, \qquad 1 \le i \le I. \tag{15}$$

Lorsque I dépasse $\operatorname{card}\mathcal{B}$, on peut résoudre le système de congruences (15), et en déduire la valeur de $\log_g p$ pour $p \in \mathcal{B}$. Notons que la résolution du système de congruences (15) est un peu plus délicate que celle d'un système linéaire.

Le deuxième temps consiste à calculer le logarithme discret de x. Pour $k = 1, 2, 3, \ldots$, on calcule $(g^k x \bmod P)$ jusqu'à ce que l'on trouve une valeur de k telle que cette quantité se factorise dans la base $\mathcal{B}$. On en déduit alors la valeur de $\log_g x$. Avec quelques améliorations, le temps de calcul est en $L^{1+o(1)}$ où $L = L(P)$ est défini par

(7). On remarque que ce temps est comparable au temps d'exécution des algorithmes de factorisation du crible quadratique ou des courbes elliptiques, avec N remplacé par P.

Soit $q = P^n$ une puissance de nombre premier. On sait que le groupe multiplicatif $\mathbb{F}_q^*$ des éléments non nuls du corps $\mathbb{F}_q$ à q éléments est cyclique: si g en est un générateur, on peut définir le logarithme discret en base g. La méthode de calcul du logarithme discret détaillée ci-dessus dans le cas $n = 1$, $q = P$ s'étend au cas $n > 1$ (cf. [17]). La méthode de factorisation du crible de corps de nombres s'est aussi étendue à ce problème (cf. [26]), fournissant un algorithme de calcul du logarithme discret dont le temps d'exécution est en général, avec la notation (14), de $L_q(1/3, c)$ pour une constante c convenable.

Soit $\mathcal{E}_{a,b}(P)$ une courbe elliptique modulo P. On sait que le groupe des points de cette courbe est soit cyclique soit produit de deux groupes cycliques (cf. [6]). Lorsque ce groupe est cyclique, on peut utiliser les méthodes de cryptographie exposées en 5(c). Pour le moment, il n'existe pas d'algorithme sous-exponentiel (c'est-à-dire en $O(P^{o(1)})$) pour le calcul du logarithme discret sur une courbe elliptique. C'est pourquoi, on considère très sérieusement la possibilité d'utiliser des protocoles cryptographiques du type 5(c) sur une courbe elliptique avec des valeurs de P de 60 chiffres décimaux, alors que les protocoles de type RSA nécessitent des nombres n d'au moins 200 chiffres.

8 Quelques autres exemples d'utilisation des ordinateurs en arithmétique

(a) *Factorisation des polynômes sur* $\mathbb{Q}$. Soit un polynôme à coefficients rationnels. Comment trouver ses facteurs irréductibles? Dans l'article [13] (cf. aussi [6] ou [15]), les frères Lenstra et Lovász inventaient l'algorithme LLL (cf. 2(g)) pour résoudre ce problème en temps polynomial par rapport au degré.

(b) *Calcul des décimales de* π. Le record actuel est détenu par Y. Kanada qui a calculé en 1997 plus de 50 milliards de décimales de π. En utilisant la formule démontrée en 1995 par Plouffe, Bailey et P. Borwein:

$$\pi = \sum_{n=0}^{\infty} \frac{1}{16^n} \left(\frac{4}{8n+1} - \frac{2}{8n+4} - \frac{1}{8n+5} - \frac{1}{8n+6} \right),$$

on peut déterminer un chiffre binaire de π sans avoir à calculer les précédents. C'est ainsi que F. Bellard a pu calculer en 1997 le 10^{12}-ème chiffre de π en base 2.

(c) *Calcul de* $\pi(x)$. Soit $\pi(x)$ le nombre de nombres premiers $\leq x$. Meissel en 1870 avait donné une formule de crible permettant le calcul exact de $\pi(x)$ (sans énumérer les nombres premiers) et Lehmer en 1959 avait utilisé la formule de Meissel pour calculer $\pi(10^{10})$. En 1985, Lagarias, Miller et Odlyzko, en améliorant considérablement la méthode, calculait $\pi(4 \cdot 10^{16})$. En 1993, Deléglise et Rivat calculaient $\pi(10^{18})$. Le record actuel est tenu par Deléglise avec le calcul de $\pi(10^{20}) = 2220819602560918840$ (cf. [22]).

(d) *Le calcul des zéros de la fonction* ζ *de Riemann.* La fonction ζ est définie pour $\Re s > 1$ par

$$\zeta(s) = \sum_{n=1}^{\infty} \frac{1}{n^s},$$

et on sait qu'elle se prolonge dans tout le plan complexe (sauf en $s = 1$) en une fonction analytique qui a de nombreux zéros dans la bande $0 < \Re s < 1$. L'hypothèse de Riemann, toujours non démontrée, est que ces zéros ont tous pour partie réelle $1/2$. Les calculs faits par van de Lune, te Riele et Winter (cf. [14]) montrent que dans le rectangle $0 < \Re s < 1, 0 < \Im s < 545439823.215$, il y a plus d'un milliard et demi de zéros (exactement 1500000001) tous simples et de partie réelle $1/2$.

(e) *Les calculs en théorie algébrique des nombres.* Soit $P(X)$ un polynôme irréductible sur $\mathbb{Q}$, et soit K l'extension de $\mathbb{Q}$ déterminée par P. Il existe maintenant de bons algorithmes pour calculer les diverses caractéristiques de K: nombre de classes de l'anneau des entiers, unités fondamentales, groupe de Galois, décomposition d'un idéal en produit d'idéaux premiers, etc. On en trouvera une bonne description dans [6].

(f) *Calcul du nombre de points d'une courbe elliptique modulo un nombre premier.* Nous avons vu en 4(d) et en 6(d) que les courbes elliptiques intervenaient dans un test de primalité et dans une méthode de factorisation. C'est donc un problème intéressant de calculer $E_{a,b}(p)$ le nombre de points de la courbe $y^2 = x^3 + ax + b$ modulo p. Nous avons dit en 4(d) que l'algorithme de Schoof n'était pas utilisable en pratique. Grâce à des idées nouvelles apportées par Atkin et Elkies, il est possible de calculer $E_{a,b}(p)$ pour des nombres premiers de plusieurs centaines de chiffres. Le record actuel est détenu par Morain avec 500 chiffres.

(g) *Le problème de Waring.* Balasubramanian, Deshouillers et Dress ont démontré en 1986 que tout nombre entier est somme d'au plus 19 puissances quatrièmes. En 1996, à la suite d'une série de calculs sur ordinateurs, Deshouillers, Landreau et Hennecart ont conjecturé que le nombre $N = 7373170279850$ est le plus grand nombre qui n'est pas somme de 4 cubes, autrement dit que tout nombre supérieur à N est somme d'au plus 4 cubes.

Références

[1] L.M. Adleman, C. Pomerance and R.S. Rumely. On distinguishing prime numbers from composite numbers, *Ann. of Math.* **117** (1983), 173–206.

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

[3] W.R. Alford, A. Granville and C. Pomerance. There are infinitely many Carmichael numbers, *Ann. of Math.* **139** (1994), 703–722.

[4] A.O.L. Atkin and F. Morain. Elliptic curves and primality proving. *Math. Comp.* **61** (1993), 29–68.

[5] E.R. Canfield, P. Erdős and C. Pomerance. On a problem of Oppenheim concerning factorisatio numerorum, *J. Number Theory* **17** (1983), 1–28.

[6] H. Cohen. *A course in algorithmic algebraic number theory*, Graduate Texts in Mathematics vol. 138, Springer-Verlag 1993.

[7] N.G. de Bruijn. On the number of positive integers $\leq x$ and free of prime factors $> y$, II. *Indag. Math.* **28** (1966), 239–247.

[8] A. Hildebrand and G. Tenenbaum. Integers without large prime factors, *J. Théorie des Nombres de Bordeaux* **5** (1993), 411–484.

[9] D.E. Knuth. *The Art of Computer Programming*, vol. 2, *Seminumerical Algorithms* 2nd ed., Addison Wesley, 1981.

[10] N. Koblitz. *A course in number theory and cryptography*, Graduate Texts in Mathematics vol. 114, Springer-Verlag, 1987.

[11] H.W. Lenstra, Jr. Factoring integers with elliptic curves. *Ann. of Math.* **126** (1987), 649–673.

[12] A.K. Lenstra and H.W. Lenstra, Jr. *The developpement of the number field sieve*. A.K. Lenstra and H.W. Lenstra, Jr. Eds., Lecture Notes in Math. **1554** (1993), Springer-Verlag.

[13] A.K. Lenstra, H.W. Lenstra, Jr. and L. Lovász. Factoring polynomials with rational coefficients. *Math. Ann.* **261** (1982), 515–534.

[14] J. van de Lune, H.J.J. te Riele and D.T. Winter. On The Zeros of the Riemann Zeta Function in the Critical Strip IV. *Math. Comp.* **46** (1986), 667–681.

[15] M. Mignotte. *Mathématiques pour le calcul formel*, Presses Universitaires de France, Paris 1989.

[16] J.-L. Nicolas. Tests de primalité. *Expo. Math.* **2** (1984), 223–234.

[17] A. Odlyzko. Discrete logarithms in finite fields and their cryptographic significance. Advances in Cryptology-Eurocrypt '84 (T. Beth, N. Cot and I. Ingemarsson, eds.). *Lecture Notes in Computer Science* vol. 209, 1985, Springer-Verlag, 224–314.

[18] J. Pollard. Theorems on Factorization and Primality Testing. *Proc. Camb. Phil. Soc.* **76** (1974), 521–528.

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

[20] C. Pomerance, J.L. Selfridge and S.S. Wagstaff, Jr. The pseudoprimes to $25 \cdot 10^9$. *Math. Comp.* **35** (1980), 1003–1026.

[21] C. Pomerance. A Tale of Two Sieves. *Notices of the A.M.S.* **43** (1996), 1473–1485.

[22] P. Ribenboim. *The New Book of Prime Number Records*, 3rd ed., Springer-Verlag, 1996.

[23] G. Robin. *Algorithmique et cryptographie*, Mathématiques et Applications vol. 8, Ellipses, Paris, 1991.

[24] B. Schneier. *Applied Cryptography*, 2nd ed., J.Wiley and sons, New-York, 1996.

[25] D. Shanks. Class number, a theory of factorization, and genera. In *Proc. Symp. Pure Math.* vol. **20** (1971), AMS, 415–440.

[26] O. Schirauker, D. Weber and T. Denny. Discrete logarithm: the effectiveness of the index calculus method, *Algorithmic Number Theory Symp.*, H. Cohen Ed., Proc. of the Second Int. Symp. ANTS II, Talence, France, May 1996. *Lecture Notes in Computer Science*, vol. **1122**, Springer-Verlag, 1996.

[27] D. Wiedemann. Solving sparse linear equations over finite fields. *IEEE Trans. Information Theory* **32** (1986), 54–62.

Institut Girard Desargues
Université Claude Bernard Lyon 1
43 Boulevard du 11 Novembre 1918
F-69622 Villeurbanne Cedex