Université de Limoges
Cours de D.E.A.

Théorie des Nombres & Cryptographie

François Arnault

27 janvier 2005

François ARNAULT
Université de Limoges, Faculté des Sciences & Techniques
Laboratoire d'Arithmétique de Calcul formel et d'Optimisation (UMR 6090)
123, av Albert Thomas, 87060 Limoges Cedex FRANCE
E-Mail : arnault@unilim.fr

Table des matières

■ **Introduction** 1

Chapitre 1

■ **Cryptographie classique** 3
1. — Exemples de systèmes cryptographiques 3
2. — Cryptanalyse 5
3. — Exercices 7

Chapitre 2

■ **Les standards de chiffrement par blocs** 9
1. — La description de DES 9
2. — Exemples et exercices 11
3. — Cryptanalyse de DES 13
4. — Triple-DES 14
5. — Rijndael 14
6. — Modes opératoires 15
7. — Sécurité des schémas de Feistel 17

Chapitre 3

■ **Introduction à la Théorie de l'Information** 21
1. — Rappels de Théorie des Probabilités 21
2. — Paradoxe des anniversaires 23
3. — Eléments de Théorie de l'Information 24
4. — Sécurité parfaite 27

Chapitre 4

■ **Fonctions de hachage** 29
1. — Construction de fonctions de hachage 29
2. — Fonctions de hachage classiques 30

Chapitre 5

■ **Le groupe $(\mathbb{Z}/n\mathbb{Z})^*$** 33
1. — Structure du groupe $(\mathbb{Z}/n\mathbb{Z})^*$ 33
2. — Carrés modulo un nombre premier 36
3. — Carrés modulo un entier quelconque 38

Chapitre 6

■ **Un peu plus d'arithmétique** **43**
1. — Corps finis 43
2. — Corps quadratiques 43
3. — Formes quadratiques binaires 45
4. — Fractions continues 48
5. — Suites de Lucas 52
6. — Courbes elliptiques 53

Chapitre 7

■ **Notions de Théorie de la Complexité** **57**
1. — Machines de Turing 57
2. — Extensions du concept de machine de Turing 59
3. — Décidabilité 59
4. — Complexité des algorithmes déterministes 61
5. — Complexité des algorithmes probabilistes 64

Chapitre 8

■ **Arithmétique pratique** **67**
1. — L'anneau $\mathbb{Z}$ 67
2. — L'anneau $\mathbb{Z}/m\mathbb{Z}$ 68
3. — Calcul du symbole de Jacobi 70
4. — Corps finis 70
5. — Racines carrées 73

Chapitre 9

■ **Générateurs aléatoires** **75**
1. — Générateurs cryptographiquement sûrs 75
2. — Le générateur BBS 77

Chapitre 10

■ **Chiffrement à clé publique** **81**
1. — Le cryptosystème RSA 81
2. — Le cryptosystème de Rabin 85
3. — Le cryptosystème El Gamal 86
4. — Le cryptosystème XTR 87
5. — Cryptosystèmes basés sur les codes correcteurs 90
6. — Cryptosystèmes utilisant les courbes elliptiques 91
7. — Echanges de clés 92

Chapitre 11

■ **Compléments concernant la sécurité** — **95**
1. — Sécurité sémantique — 95
2. — Optimal Asymmetric Encryption Padding (OAEP) — 96
3. — Le status du problème de Diffie-Hellman — 97

Chapitre 12

■ **Signature, authentification** — **101**
1. — Signature RSA — 101
2. — Signatures El Gamal, DSS et EC-DSA — 102
3. — Fonctions de hachage en signature — 103

Chapitre 13

■ **Identification** — **107**
1. — Protocoles à une passe — 107
2. — Protocoles à deux passes — 107
3. — Protocoles sans divulgation d'information (*Zero-knowledge*) — 108
4. — Engagement — 111
5. — Kerberos — 112

Chapitre 14

■ **Primalité** — **115**
1. — Tests de primalité — 115
2. — Preuves de Primalité — 118
3. — Un algorithme déterministe polynomial — 121
4. — Génération aléatoire de clés cryptographiques — 124

Chapitre 15

■ **Factorisation** — **129**
1. — Les méthodes classiques — 129
2. — Les méthodes à combinaison de congruences — 133

Chapitre 16

■ **Logarithme discret** — **139**
1. — Méthodes élémentaires — 139
2. — Méthodes sous-exponentielles — 142

■ **Bibliographie** — **145**

Introduction

L'histoire de la Cryptographie est déjà longue. On rapporte son utilisation en Egypte il y a 4000 ans. Toutefois, pendant des siècles, les méthodes utilisées étaient restées souvent très primitives. D'autre part, sa mise en œuvre était limitée aux besoins de l'armée et de la diplomatie.

Les méthodes de chiffrement et de cryptanalyse ont connu un développement très important au cours de la seconde guerre mondiale et ont eu une profonde influence sur le cours de celle-ci.

Mais c'est la la prolifération actuelle des systèmes de communication qui a fait sortir la cryptographie du domaine militaire. De plus, elle a diversifié la demande et provoqué le développement de nouvelles techniques cryptographiques. Elle est à l'origine d'un développement rapide depuis les dernières décennies, qui ne semble pas s'essouffler aujourd'hui, bien au contraire.

Les principaux services offerts par la cryptographie moderne sont les suivants :

• Confidentialité. Assurer que les données concernées ne pourront être dévoilées qu'aux personnes autorisées.

• Intégrité. Assurer que les données ne seront pas altérées (intentionnellement ou non) pendant leur transmission ou leur stockage.

• Authentification. Prouver l'origine d'une donnée, ou la validité d'un objet cryptographique (clé, moyen de paiement, etc).

• Identification. Prouver qu'un objet cryptographique ou qu'une personne est exactement celui/celle qu'il/elle prétend être.

• Signature proprement dite (*undeniability — non-repudiation*). Permet à une personne de prendre part à un contrat avec impossibilité de renier ensuite ses engagements.

— ⋆ — ⋆ — ⋆ — ⋆ — ⋆ —

Quelques repères historiques de la cryptographie moderne :

• **1975.** Conception de DES, standard de chiffrement de données, adopté en 1977.

• **1976.** Article de Diffie & Hellman introduisant l'idée de système à clé publique.

• **1978.** Invention de RSA, le premier système concret de cryptographie à clé publique.

• **1985.** Invention du système cryptographique El Gamal.

• **1988.** Invention du premier algorithme de factorisation de type NFS par Pollard.

• **1991.** Adoption du premier standard de signature, ISO 9796, basé sur RSA.

• **1994.** Adoption de DSS, standard de signature basé sur le logarithme discret.

• **1998.** Publication du standard PKCS#1 v2.0, adoptant une variante de OAEP pour assurer la sécurité du chiffrement RSA.

• **2000.** Adoption de Rijndael, comme AES (successeur du DES).

Notations

Pour $a, b \in \mathbb{Z}$ avec $b > 0$, on note $a \bmod b$ le reste de la division euclidienne de a par b. C'est un entier de l'intervalle $[0, b-1]$. Ne pas confondre avec la **relation** $a \equiv b \pmod{n}$.

Nous utiliserons constamment l'anneau quotient $\mathbb{Z}/n\mathbb{Z}$ et son groupe des inversibles $(\mathbb{Z}/n\mathbb{Z})^*$. Parfois, nous identifierons la classe modulo n d'un entier a avec l'entier $a \bmod n$ (parce que la plupart des algorithmes représentent la classe de a par l'entier $a \bmod n$). Cela revient à identifier $\mathbb{Z}/n\mathbb{Z}$ avec l'ensemble $\{0, 1, \ldots, n-1\}$ muni de l'addition et de la multiplication modulaires, que nous noterons $\mathbb{Z}_n$. Dans le même état d'esprit, $\mathbb{Z}_n^*$ désignera le même ensemble $\mathbb{Z}_n$ privé des entiers x tels que $\mathrm{pgcd}(x, n) > 1$ et muni de la multiplication modulaire.

Pour certains termes, on donnera la traduction anglaise en caractères penchés (*slanted type*).

Les références aux résultats, définitions, exercices, etc, sont données, à l'intérieur d'un même chapitre, sur deux chiffres `y.z` où `y` est le numéro de section. Les références externes au chapitre sont sous la forme `x.y.z` où l'on a rajouté le numéro `x` du chapitre dans lequel est définie la référence. Pour les sections, figures et équations, les références sont données sous la forme `y` ou `x.y` selon qu'il s'agit de références internes ou externes au chapitre.

Chapitre 1

Cryptographie classique

Ce chapitre est extrait du livre de Stinson [127]. Comme autres références, on peut citer par exemple les livres de Beker & Piper [16], de Denning [40], de Kahn [54], et de Konheim [62]. De plus, un livre de Turing [129] est consacré à la machine Enigma.

Alice veut envoyer un message à Bob. Mais elle ne veut pas que quelqu'un d'autre (par exemple Oscar) prenne connaissance du contenu de ce message. Alice va utiliser un système cryptographique pour que le message, s'il est intercepté par Oscar, ne soit pas intelligible pour lui. Seul le destinataire du message sera capable de retrouver le contenu du message, grâce à une information supplémentaire — la clé — dont il aura convenu à l'avance avec Alice.

L'information que veut transmettre Alice peut prendre de nombreuses formes : un texte, une donnée numérique, etc. Alice va tout d'abord découper et convertir le message en une suite de blocs $(x_i)_{1\leqslant i\leqslant n}$ appartenant tous à un ensemble fini $\mathcal{P}$ (l'ensemble des lettres de l'alphabet par exemple). Elle va ensuite utiliser une fonction de chiffrement e_k associée à la clé k, $e_k : \mathcal{P} \to \mathcal{C}$ où $\mathcal{C}$ est un ensemble fini, pour calculer $y_i = e_K(x_i)$ $(1 \leqslant i \leqslant n)$. Elle transmet alors les textes chiffrés y_i à Bob, qui recalculera les $x_i = d_k(y_i)$ en utilisant une fonction de déchiffrement $d_k : \mathcal{C} \to \mathcal{P}$.

De manière formelle :

0.1. — Définition. Un *système cryptographique* est la donnée de

• Un ensemble fini $\mathcal{P}$ appelé l'espace des textes clairs (*plaintexts*).

• Un ensemble fini $\mathcal{C}$ appelé l'espace des textes chiffrés (*ciphertexts*).

• Un ensemble fini $\mathcal{K}$ appelé l'espace des clés (*keys*).

• Pour chaque clé $k \in \mathcal{K}$, une fonction de chiffrement (*encryption*) $e_k : \mathcal{P} \to \mathcal{C}$ et une fonction de déchiffrement (*decryption*) $d_k : \mathcal{C} \to \mathcal{P}$ telles que $d_k \circ e_k = \mathrm{Id}_{\mathcal{P}}$.

1. Exemples de systèmes cryptographiques

La plupart des exemples de cette section sont trop simples pour offrir la moindre sécurité. Leur intérêt est purement pédagogique.

Identifions l'alphabet usuel avec $\mathbb{Z}_{26}$ par $a = 0, b = 1, \ldots, z = 25$. On a alors les systèmes cryptographiques suivants :

1.1. — Chiffrement par décalage. $\mathcal{P} = \mathcal{C} = \mathcal{K} = \mathbb{Z}_{26}$. On a $e_k(x) = x + k$ et $d_k(y) = y - k$. C'est un cas particulier des deux suivants.

Exemple, pour $k = 3$, le texte clair `cryptographie` est chiffré en `FUBSWRJUDSKLH`.

Ce système de chiffrement n'est pas sûr du tout puisque l'espace des clés ne contient que 26 éléments. On peut facilement les essayer une à une jusqu'à trouver la bonne.

1.2. — Chiffrement par substitution. $\mathcal{P} = \mathcal{C} = \mathbb{Z}_{26}$. L'espace des clés est l'ensemble des permutations de $\mathbb{Z}_{26}$. Pour $k \in \mathcal{K}$, on a $e_k = k$ et $d_k = k^{-1}$.

Par exemple, pour la permutation suivante

a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w	x	y	z
P	O	M	L	I	K	U	J	N	Y	H	B	T	G	R	F	V	D	C	E	Z	S	X	A	Q	W

le texte clair `cryptographie` est chiffré en `MDQFERUDPFJNI`.

1.3. — Chiffrement de Vigenère (1586). $\mathcal{P} = \mathcal{C} = \mathcal{K} = \mathbb{Z}_{26}^m$ ($m \in \mathbb{N}^*$). Soit $k = (k_1, \ldots, k_m) \in \mathcal{K}$ alors $e_k(x_1, \ldots, x_m) = (x_1 + k_1, \ldots, x_m + k_m)$ et $d_k(y_1, \ldots, y_m) = (y_1 - k_1, \ldots, y_m - k_m)$.

Exemple, pour $m = 3$ et $k = \texttt{cle} = (2, 11, 4)$, le texte clair `cryptographie` est chiffré en `ECCRESICERSMG`.

1.4. — Masque jetable (*one time pad*). Identique au chiffrement de Vigenère mais on n'utilise la clé qu'une seule fois, ce qui contraint à choisir une clé aussi longue que le message à transmettre.

Exemple, considérons le texte clair `cryptographie` $= (2, 17, 24, 15, 19, 14, 6, 17, 0, 15, 7, 8, 4)$ et appliquons le chiffrement par masque jetable avec la clé `algorithmique` $= (0, 11, 6, 14, 17, 8, 19, 7, 12, 8, 16, 20, 4)$. On obtient alors le message chiffré $(2, 2, 4, 3, 10, 22, 25, 24, 12, 23, 23, 2, 8) =$ `CCEDKWZYMXXCI`.

1.5. — Chiffrement de Hill. $\mathcal{P} = \mathcal{C} = \mathbb{Z}_{26}^m$ ($m \in \mathbb{N}^*$). L'espace des clés est l'ensemble $\mathrm{GL}_m(\mathbb{Z}_{26})$ des matrices inversibles d'ordre m à coefficients dans $\mathbb{Z}_{26}$. Soit $k \in \mathcal{K}$, On a alors $e_k(x_1, \ldots, x_m) = (x_1, \ldots, x_m)k$ et $d_k(y_1, \ldots, y_m) = (y_1, \ldots, y_m)k^{-1}$.

Exemple, pour $m = 2$ et $k = \left(\begin{smallmatrix} 4 & 3 \\ 9 & 7 \end{smallmatrix}\right)$ le texte clair `cryptographe` est codé numériquement par

$$\big((2, 17), (24, 15), (19, 14), (6, 17), (0, 15), (7, 4)\big).$$

Il est alors chiffré en

$$\big((5, 21), (23, 21), (20, 25), (21, 7), (5, 1), (12, 23)\big)$$

ce qui correspond finalement à `FVXVUZVHFBMX`.

1.6. — Exercice. Combien y a-t-il d'éléments dans $\mathrm{GL}_2(\mathbb{F}_p)$ pour p premier ? Combien y a-t-il d'éléments dans $\mathrm{GL}_2(\mathbb{Z}_{26})$?

1.7. — Chiffrement par permutation. $\mathcal{P} = \mathcal{C} = \mathbb{Z}_{26}^m$ ($m \in \mathbb{N}^*$) et $\mathcal{K}$ est l'ensemble des permutations de $\mathbb{Z}_m$. Pour $k \in \mathcal{K}$, on a alors $e_k(x_1, \ldots, x_m) = (x_{k(1)}, \ldots, x_{k(m)})$ et $d_k(y_1, \ldots, y_m) = (y_{k^{-1}(1)}, \ldots, y_{k^{-1}(m)})$. On peut voir ce système comme un cas particulier du chiffrement de Hill.

Exemple, si $m = 6$ et k est la permutation qui envoie $(1, 2, 3, 4, 5, 6)$ sur $(4, 2, 1, 5, 6, 3)$ alors le texte clair `cryptographe` est chiffré en `PRCTOYPRGHEA`.

1.8. — Chiffrement en chaîne. $\mathcal{P} = \mathcal{C} = \mathbb{Z}_{26}^m$ ($m \in \mathbb{N}^*$). La clé $k = (k_1, \ldots, k_m)$ est un élément de $\mathbb{Z}_{26}^m$ donné par sa première composante k_1 et par une règle qui permet de calculer chaque k_i ($i > 1$) en fonction de k_{i-1} et $x_1, \ldots, x_{i-1}$. La portion de clé k_i servira alors à chiffrer x_i ; par exemple, on pourra poser $y_i = x_i + k_i$. Si la suite des k_i est périodique, on parle de chiffrement en chaîne périodique et c'est une généralisation du chiffrement de Vigenère.

Exemple, si $m = 13$, $k_1 = 4$ et $k_i = x_{i-1} + k_{i-1} = y_{i-1}$, alors le texte clair `cryptographie` est codé numériquement en $(2, 17, 24, 15, 19, 14, 6, 17, 0, 15, 7, 8, 4)$. Ces valeurs sont chiffrées en $(6, 23, 21, 10, 3, 17, 23, 4, 4, 19, 0, 8, 12)$, la clé complète calculée au fur et à mesure étant $(4, 6, 23, 21, 10, 3, 17, 23, 4, 4, 19, 0, 8)$. En caractères alphabétiques, le message chiffré est `GXVKDRXEETAIM`.

1.9. — Surchiffrement. C'est la composition de deux méthodes de chiffrement. Le message chiffré par la première méthode est utilisé comme un "message clair" pour la deuxième méthode.

1.10. — Exercice. Etudier la composition de deux chiffrements de Vigenère. Comment composer plusieurs chiffrements de Vigenère pour obtenir une période finale maximale, en utilisant des clés dont la somme des longueurs est au plus 19 ?

La machine *Enigma*

La machine *Enigma* inventée en Allemagne par Scherbius et qui a joué un rôle important lors de la deuxième guerre mondiale était une machine à rotors (des disques enfilés le long d'un axe commun et qui peuvent tourner séparément autour de cet axe). Chaque rotor était muni de 26 contacts sur chaque face (un pour chaque lettre) et était câblé de façon interne réalisant ainsi une permutation de $\mathbb{Z}_{26}$. Cette permutation changeait (en une permutation conjuguée) lorsque le rotor tournait. Enigma fonctionnait (dans sa première version) avec trois rotors accolés ce qui revenait à composer les trois permutations. Elle possédait en tout cinq rotors, ce qui permettait de varier le groupe de trois rotors utilisés pour chaque transmission.

A chaque caractère, les rotors tournaient de façon incrémentale (à la manière d'un compteur), le premier entraînant le deuxième d'un cran à chaque tour et le deuxième entraînant le troisième et lui-même d'un cran à chaque tour. On obtenait ainsi un chiffrement en chaîne périodique de période $26 \times 25 \times 26 = 16900$.

En plus, une permutation supplémentaire était réalisée par un tableau à connecteurs. La clé était donc constituée de la connaissance des rotor choisis (dans l'ordre), de la position initiale de chacun d'eux et de la configuration du tableau.

On trouve un simulateur de la machine Enigma à l'URL suivante :
`http://www.iro.umontreal.ca/~crepeau/CRYPTO/ENIGMA/ENIGMA/enigma.html`.

2. Cryptanalyse

C'est l'étude des systèmes cryptographiques, en particulier de leurs faiblesses, dans le but de retrouver des messages clairs correspondant à des messages chiffrés dont on n'est pas destinataire.

Principe de Kerckhoffs

2.1. — Principe de Kerckhoffs. Un principe fondamental de la cryptographie a été énoncé par Kerckhoffs à la fin du dix-neuvième siècle. Il exprime que la méthode de chiffrement utilisée doit "pouvoir tomber sans inconvénient aux mains de l'ennemi". Autrement dit, la sécurité d'un chiffrement doit reposer uniquement sur la protection de la clé.

Ce principe a plusieurs justifications. Principalement :

• La confidentialité d'un algorithme est difficile à garantir. Il est en général connu de plusieurs personnes (parfois nombreuses) et il est souvent diffusé dans des logiciels ou dispositifs hardware à des utilisateurs non habilités au secret. La confidentialité de l'algorithme peut succomber à la corruption ou au *reverse engineering*.

• La sécurité d'un algorithme secret est difficile à évaluer (nombre d'algorithmes à l'origine secrets se sont révélés extrèmement faibles). Il est généralement admis que la meilleure garantie de sécurité d'un algorithme est apportée par une longue période d'évaluation par la communauté cryptographique mondiale.

• Un algorithme secret peut dissimuler des propriétés indésirables pour l'utilisateur final (existence de clés faibles, par exemple). Il n'est donc pas adapté si la confiance envers le concepteur n'est pas établie.

• Enfin, pour le théoricien, c'est une hypothèse de travail sans laquelle il est impossible d'obtenir des résultats rigoureux de sécurité.

Différents types de cryptanalyse

Un *attaquant* est donc une personne qui tente de *décrypter* des messages, c'est-à-dire de retrouver des clairs à partir de chiffrés sans connaître la clé. On réserve généralement le verbe *déchiffrer* à l'action du destinataire légitime qui effectue l'opération inverse du chiffrement.

La cryptanalyse d'un système cryptographique peut-être :

• *Une cryptanalyse partielle.* L'attaquant découvre alors le texte clair correspondant à un ou plusieurs messages chiffrés interceptés.

• *Une cryptanalyse totale.* L'attaquant découvre un moyen de déchiffrer tous les messages, aussi bien ceux qu'il a interceptés que ceux à venir, par exemple en découvrant la clé utilisée.

Selon les moyens dont dispose l'attaquant, on distingue plusieurs types d'attaques. Par ordre de moyens croissants on a :

• *Attaque à messages chiffrés (seulement).* L'attaquant a seulement la possibilité d'intercepter un ou plusieurs messages chiffrés.

• *Attaque à messages clairs.* L'attaquant dispose d'un ou plusieurs messages clairs avec les messages chiffrés correspondants.

• *Attaque à messages clairs choisis.* L'attaquant a la possibilité d'obtenir la version chiffrée de messages clairs de son choix. On distingue alors deux sous-types d'attaque, suivant que l'attaquant est contraint de choisir les clairs en une seule fois, ou au contraire peut faire évoluer ses choix au fur et à mesure des résultats obtenus. Dans le deuxième cas, on parle d'attaque *adaptative* à messages clairs choisis.

• *Attaque à messages chiffrés choisis.* L'attaquant a temporairement l'opportunité de déchiffrer les messages de son choix (en ayant accès par exemple à une machine déchiffrante). Il tente alors d'en profiter pour obtenir des informations lui permettant de décrypter ensuite d'autres messages par ses propres moyens. Comme dans le point précédent, on peut distinguer deux sous-types : attaque adaptative ou non.

Fréquences des lettres

	Anglais	Français		Anglais	Français
a	8.17	8.25	n	6.75	7.25
b	1.49	1.25	o	7.51	5.75
c	2.78	3.25	p	1.93	3.75
d	4.25	3.75	q	0.10	1.25
e	12.70	17.75	r	5.99	7.25
f	2.23	1.25	s	6.33	8.25
g	2.02	1.25	t	9.06	7.25
h	6.09	1.25	u	2.76	6.25
i	6.97	7.25	v	0.98	1.75
j	0.15	0.75	w	2.36	0.00
k	0.77	0.00	x	0.15	0.00
l	4.03	5.75	y	1.97	0.75
m	2.41	3.25	z	0.07	0.00

Table 1. Fréquences des lettres en Anglais et en Français (en %)

Le chiffrement par substitution est attaquable en examinant les fréquences d'apparition de chaque lettre. Les fréquences moyennes relevées dans un texte après suppression des espaces et de la ponctuation sont indiquées dans la table 1. En comparant avec les fréquences observées dans le texte chiffré, on peut retrouver suffisamment d'indications pour le décrypter. On peut aussi s'aider des fréquences d'apparition de chaque digramme ou trigramme (deux ou trois lettres consécutives). Par exemple, les digrammes les plus courants en anglais sont (par ordre de fréquence décroissante) TH, HE, IN, ER, AN, RE, ED, ON, ES, ST, EN, AT, TO, NT, HA, ND, OU, EA, NG, AS, OR, TI, IS, ET, IT, AR, TE, SE, HI et OF. Quant aux trigrammes, ce sont THE, ING, AND, HER, ERE, ENT, THA, NTH, WAS, ETH, FOR et DTH.

Indices de coïncidence, de coïncidence mutuelle

L'indice de coïncidence d'un message x de longueur n est la proportion, parmi les $n(n-1)/2$ paires de lettres de x, de paires de lettres identiques. Si f_i est le nombre d'occurences de la lettre i dans x alors l'indice de coïncidence est donné par

$$I_c(x) = \frac{1}{n(n-1)} \sum_{i=\mathtt{a}}^{\mathtt{z}} f_i(f_i - 1).$$

Si chaque lettre de l'alphabet apparaissait avec à peu près la même fréquence dans un message alors son indice de coïncidence serait proche de $1/26 \simeq 0.0385$. Mais les lettres d'un texte réel ont des fréquences différentes et l'indice de coïncidence est en général plus grand, proche de la valeur $\sum_{i=\mathtt{a}}^{\mathtt{z}} p_i^2$, où p_i est la fréquence moyenne de la lettre i. Les valeurs standards de l'indice de coïncidence pour différentes langues sont données par la table suivante.

Anglais	Français	Allemand	Portugais	Espagnol	Russe
0.066	0.076	0.076	0.079	0.078	0.053

Table 2. Indices de coïncidence standards dans plusieurs langues

L'indice de coïncidence mutuelle de deux messages x et x' de longueurs respectives n et n' est la proportion, parmi les nn' couples formés d'une lettre de x et d'une lettre de x', de couples de lettres identiques. Si f_i et f'_i sont les nombres respectifs d'occurences de la lettre i dans x et x' alors l'indice de coïncidence mutuelle est donné par

$$I_c(x, x') = \frac{1}{nn'} \sum_{i=\mathtt{a}}^{\mathtt{z}} f_i f'_i.$$

Si x et x' sont deux messages écrits dans la même langue, l'indice de coïncidence mutuelle est proche des valeurs données par la table 2. Cela est encore vrai si on applique à x et x' un chiffrement par décalage ou substitution utilisant la même clé. Par contre, si on utilise deux clés de décalage distinctes, l'indice de coïncidence mutuelle prend alors des valeurs plus basses, de 0.031 à 0.045 (pour un texte en anglais).

Application au cryptosystème de Vigenère

La longueur de la clé utilisée dans un cryptosystème de Vigenère peut être découverte de la manière suivante. Tout d'abord, Kasiski à remarqué en 1863 que si une séquence suffisamment longue (à partir de trois lettres) se retrouve à plusieurs endroits dans un texte chiffré alors elle provient probablement d'une répétition dans le texte clair, et le décalage entre les deux séquences est un multiple de la longueur de la clé. En repérant plusieurs couples de séquences identiques et en calculant le pgcd des décalages correspondants, on obtient en général la longueur de la clé.

D'autre part, supposons que l'on ait un texte c chiffré par la méthode de Vigenère avec une clé de longueur m. Considérons les m sous-messages c_i, avec $0 \leqslant i \leqslant m-1$, obtenus en prenant la sous-suite des lettres de c de rang congru à i modulo m. Les sous-messages c_i ont chacun un indice de coïncidence proche de la valeur standard. Une fois que l'on a des soupçons (par le test de Kasiski) sur la valeur de la longueur m de la clé, on peut donc obtenir confirmation en calculant ces m indices.

Un fois que l'on a déterminé la longueur m de la clé utilisée pour former un message chiffré c par la méthode de Vigenère, on peut examiner les m sous-messages $c_1, \ldots, c_m$ qui sont chacun des décalages de sous-messages du texte clair. On peut déterminer la différence entre les clés de décalages utilisées pour deux sous-messages c_i et c_j en calculant les indices de coïncidence mutuelle de c_i avec c_j^k où les c_j^k sont obtenus à partir de c_j par un décalage de clé k.

Enfin, une fois déterminés les décalages relatifs des sous-messages c_i, il ne reste plus que 26 clés possibles que l'on peut tester exhaustivement et donc retrouver le texte clair.

3. Exercices

3.1. — Exercice. Décrypter le message suivant, obtenu par substitution (c'est un texte en anglais).

```
EMGLOSUDCGDNCUSWYSFHNSFCYKDPUMLWGYICOXYSIPJCK
QPKUGKMGOLICGINCGACKSNISACYKZSCKXECJCKSHYSXCG
OIDPKZCNKSHICGIWYGKKGKGOLDSILKGOIUSIGLEDSPWZU
GFZCCNDGYYSFUSZCNXEOJNCGYEOWEUPXEZGACGNFGLKNS
ACIGOIYCKXCJUCIUZCFZCCNDGYYSFEUEKUZCSOCFZCCNC
IACZEJNCSHFZEJZEGMXCYHCJUMGKUCY
```

3.2. — **Exercice.** Décrypter le message suivant, obtenu par un chiffrement de Vigenère (c'est un texte en anglais).

```
CHREEVOAHMAERATBIAXXWTNXBEEOPHBSBQMQEQERBW
RVXUOAKXAOSXXWEAHBWGJMMQMNKGRFVGXWTRZXWIAK
LXFPSKAUTEMNDCMGTSXMXBTUIADNGMGPSRELXNJELX
VRVPRTULHDNQWTWDTYGBPHXTFALJHASVBFXNGLLCHR
ZBWELEKMSJIKNBHWRJGNMGJSGLXFEYPHAGNRBIEQJT
AMRVLCRREMNDGLXRRIMGNSNRWCHRQHAEYEVTAQEBBI
PEEWEVKAKOEWADREMXMTBHHCHRTKDNVRZCHRCLQOHP
WQAIIWXNRMGWOIIFKEE
```

Chapitre 2

Les standards de chiffrement par blocs

Nous décrivons ici les deux standards adoptés successivement par l'administration américaine et universellement utilisés dans le monde. Le premier est le *Data Encryption Standard* (DES) publié en 1975 et adopté par le *National Bureau of Standards* [91] en 1977. Le second a été publié en 1978 sous le nom de Rijndael [38] et adopté comme standard *Advanced Encryption Standard* (AES) en 2001 [94].

1. La description de DES

C'est le cryptosystème qui a été le plus utilisé et le plus étudié. Il a résisté remarquablement bien aux efforts des cryptanalystes pendant 25 ans. C'est seulement la longueur de sa clé, largement suffisante au moment de sa conception mais désormais trop courte, qui lui vaut aujourd'hui de ne plus assurer un très bon niveau de sécurité.

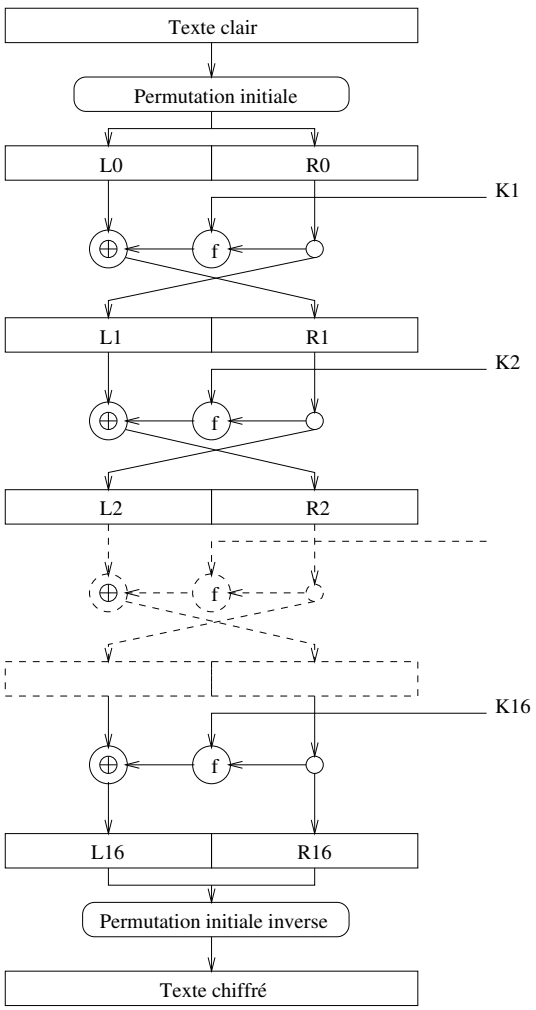

Figure 1. DES : Schéma général

Schéma général

Le DES utilise une clé K de 56 bits, pour chiffrer des blocs de 64 bits, les blocs chiffrés obtenus ayant aussi 64 bits. Le bloc de texte clair subit d'abord une permutation initiale. Puis on itère 16 fois une procédure identique décrite ci-après, où la moitié droite est recopiée telle quelle à gauche, et la moitié gauche est transmise à droite en subissant au passage une modification dépendante de la clé. A la fin, on inverse les moitiés gauches et droites (ou bien, comme sur le schéma, on supprime le croisement de la dernière étape), et on applique l'inverse de la permutation initiale pour obtenir le bloc chiffré. Le schéma général de DES représenté dans la figure 1 (on a seulement représenté quelques-unes des 16 étapes).

La première et la dernière opérations lors du chiffrement d'un bloc par DES est l'application d'une permutation initiale, et de son inverse à la fin. Cette permutation n'a aucun rôle dans la sécurité de l'algorithme. La permutation initiale et son inverse sont décrites par la figure 2. Les tableaux se lisent de gauche à droite et de haut en bas, le n-ième nombre est la position avant permutation du bit qui se trouve en n-ième position après permutation.

Permutation initiale

58	50	42	34	26	18	10	2
60	52	44	36	28	20	12	4
62	54	46	38	30	22	14	6
64	56	48	40	32	24	16	8
57	49	41	33	25	17	9	1
59	51	43	35	27	19	11	3
61	53	45	37	29	21	13	5
63	55	47	39	31	23	15	7

Permutation initiale inverse

40	8	48	16	56	24	64	32
39	7	47	15	55	23	63	31
38	6	46	14	54	22	62	30
37	5	45	13	53	21	61	29
36	4	44	12	52	20	60	28
35	3	43	11	51	19	59	27
34	2	42	10	50	18	58	26
33	1	41	9	49	17	57	25

Figure 2. La permutation initiale et son inverse

Après la permutation initiale, le message est séparé en deux moitiés de 32 bits, désignées par L_0 et R_0. A chaque itération de la procédure, on détermine deux groupes de 32 bits L_i et R_i en fonction de L_{i-1} et R_{i-1} obtenus précédemment. Pour cela, on utilise une clé intermédiaire K_i de 48 bits, calculée à partir de K et on applique les formules suivantes :

$$L_i = R_{i-1} \qquad \text{et} \qquad R_i = L_{i-1} \oplus f(R_{i-1}, K_i).$$

La fonction f

La fonction f est schématisée par la figure 3. Tout d'abord, l'argument de gauche qui possède 32 bits est expansé en 48 bits en redoublant certains bits. Cette expansion est précisée par la figure 4. Ensuite, on calcule le ou exclusif de cet argument expansé avec le deuxième argument (qui n'est autre que la clé K_i). Le résultat possède $48 = 8 \times 6$ bits et est transformé en une chaîne de $32 = 8 \times 4$ bits en utilisant des dispositifs appelés boîtes-S qui calculent un bloc de 4 bits à partir d'un bloc de 6 bits. Enfin, on applique la permutation décrite par la figure 4 à ces 32 bits pour obtenir la valeur de f.

Les boîtes-S

Il y a huit boîtes-S différentes. Elles sont représentées dans la figure 5 par des tableaux à 2 lignes et 16 colonnes. Les premiers et derniers bits de l'entrée déterminent une ligne du tableau, les autres bits déterminent une colonne. La valeur numérique trouvée à cet endroit indique la valeur des quatre bits de sortie.

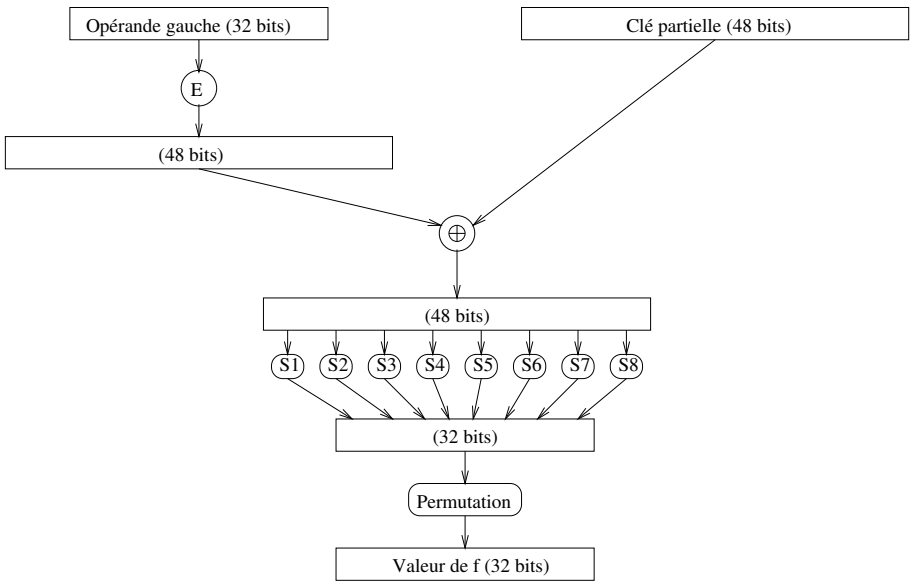

Figure 3. Schéma de la fonction f

Fonction E d'expansion

32	1	2	3	4	5	4	5	6	7	8	9
8	9	10	11	12	13	12	13	14	15	16	17
16	17	18	19	20	21	20	21	22	23	24	25
24	25	26	27	28	29	28	29	30	31	32	1

Permutation P finale

16	7	20	21	29	12	28	17
1	15	23	26	5	18	31	10
2	8	24	14	32	27	3	9
19	13	30	6	22	11	4	25

Figure 4. L'expansion du premier argument et la permutation finale de f

La diversification de la clé

Le principe de la diversification de la clé est schématisé par la figure 6. On applique d'abord une permutation PC_1 à K. Puis, à chacune des 16 étapes, chaque moitié de la chaîne de 56 bits obtenue subit une rotation à gauche, d'un cran aux étapes 1, 2, 9, 16, et de deux crans aux autres étapes. A chacune de ces étapes, on obtient une clé partielle de 48 bits en appliquant la règle d'extraction PC_2. La permutation PC_1 et la règle PC_2 sont détaillées par la figure 7. Les 56 bits de K y sont numérotés de 1 à 64 en évitant les multiples de 8, puisque dans la pratique, ces positions multiples de 8 sont utilisées pour insérer des bits de parité (la convention utilisée est fréquemment celle de la parité impaire).

2. Exemples et exercices

2.1. — Exemple. Voici un exemple de message chiffré par DES, en notation hexadécimale. Les trois blocs de message clair sont la représentation ASCII de la chaîne "Now is the time for all ".

$$x = \texttt{4E6F772069732074 68652074696D6520 666F7220616C6C20}$$
$$k = \texttt{0123456789ABCDEF}$$
$$y = \texttt{3FA40E8A984D4815 6A271787AB8883F9 893D51EC4B563B53}$$

S_1	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	14	4	13	1	2	15	11	8	3	10	6	12	5	9	0	7
1	0	15	7	4	14	2	13	1	10	6	12	11	9	5	3	8
2	4	1	14	8	13	6	2	11	15	12	9	7	3	10	5	0
3	15	12	8	2	4	9	1	7	5	11	3	14	10	0	6	13

S_2	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	15	1	8	14	6	11	3	4	9	7	2	13	12	0	5	10
1	3	13	4	7	15	2	8	14	12	0	1	10	6	9	11	5
2	0	14	7	11	10	4	13	1	5	8	12	6	9	3	2	15
3	13	8	10	1	3	15	4	2	11	6	7	12	0	5	14	9

S_3	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	10	0	9	14	6	3	15	5	1	13	12	7	11	4	2	8
1	13	7	0	9	3	4	6	10	2	8	5	14	12	11	15	1
2	13	6	4	9	8	15	3	0	11	1	2	12	5	10	14	7
3	1	10	13	0	6	9	8	7	4	15	14	3	11	5	2	12

S_4	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	7	13	14	3	0	6	9	10	1	2	8	5	11	12	4	15
1	13	8	11	5	6	15	0	3	4	7	2	12	1	10	14	9
2	10	6	9	0	12	11	7	13	15	1	3	14	5	2	8	4
3	3	15	0	6	10	1	13	8	9	4	5	11	12	7	2	14

S_5	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	2	12	4	1	7	10	11	6	8	5	3	15	13	0	14	9
1	14	11	2	12	4	7	13	1	5	0	15	10	3	9	8	6
2	4	2	1	11	10	13	7	8	15	9	12	5	6	3	0	14
3	11	8	12	7	1	14	2	13	6	15	0	9	10	4	5	3

S_6	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	12	1	10	15	9	2	6	8	0	13	3	4	14	7	5	11
1	10	15	4	2	7	12	9	5	6	1	13	14	0	11	3	8
2	9	14	15	5	2	8	12	3	7	0	4	10	1	13	11	6
3	4	3	2	12	9	5	15	10	11	14	1	7	6	0	8	13

S_7	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	4	11	2	14	15	0	8	13	3	12	9	7	5	10	6	1
1	13	0	11	7	4	9	1	10	14	3	5	12	2	15	8	6
2	1	4	11	13	12	3	7	14	10	15	6	8	0	5	9	2
3	6	11	13	8	1	4	10	7	9	5	0	15	14	2	3	12

S_8	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	13	2	8	4	6	15	11	1	10	9	3	14	5	0	12	7
1	1	15	13	8	10	3	7	4	12	5	6	11	0	14	9	2
2	7	11	4	1	9	12	14	2	0	6	10	13	15	3	5	8
3	2	1	14	7	4	10	8	13	15	12	9	0	3	5	6	11

Figure 5. Les boîtes-S

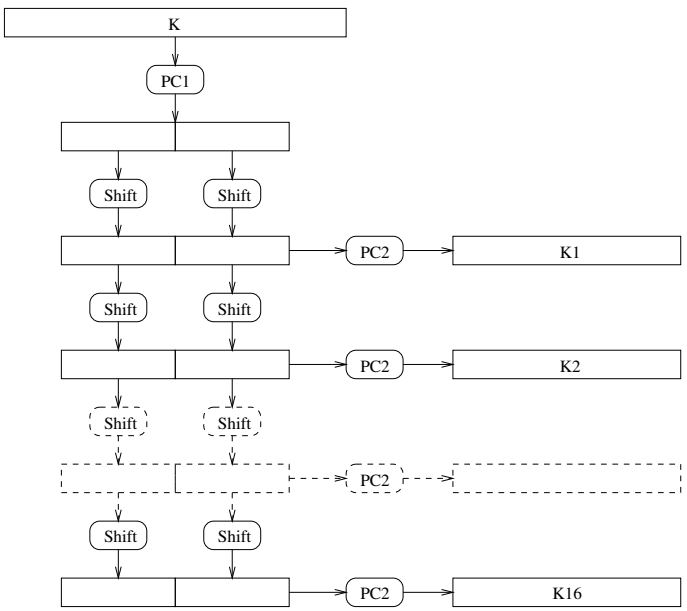

Figure 6. La diversification de la clé

2.2. — **Exercice.** On note par une barre le *non* logique bit à bit et par $\mathrm{DES}(x, k)$ le chiffrement d'un

Permutation PC_1

57	49	41	33	25	17	9
1	58	50	42	34	26	18
10	2	59	51	43	35	27
19	11	3	60	52	44	36
63	55	47	39	31	23	15
7	62	54	46	38	30	22
14	6	61	53	45	37	29
21	13	5	28	20	12	4

Règle d'extraction PC_2

14	17	11	24	1	5
3	28	15	6	21	10
23	19	12	4	26	8
16	7	27	20	13	2
41	52	31	37	47	55
30	40	51	45	33	48
44	49	39	56	34	53
46	42	50	36	29	32

Figure 7. Les permutations pour la diversification

message x par une clé k. Comparer $\mathrm{DES}(\overline{x}, \overline{k})$ et $\mathrm{DES}(x, k)$.

2.3. — Exercice. Comment déchiffrer un message chiffré par DES ? Quelle différence avec l'opération de chiffrement ?

2.4. — Exercice (clés faibles). Une clé DES est dite *faible* si les fonctions de chiffrement et de déchiffrement associées à cette clé sont égales. Montrer que les clés

`0101010101010101`, `FEFEFEFEFEFEFEFE`, `1F1F1F1F0E0E0E0E`, `0E0E0E0E1F1F1F1F`,

sont faibles.

3. Cryptanalyse de DES

Cryptanalyse différentielle

La cryptanalyse différentielle a été introduite par Biham et Shamir et ils ont étudié le cas de DES dans [19]. C'est une attaque à messages clairs choisis. Elle permet de casser des versions restreintes de DES, dans lesquels on diminue le nombre de tours de la boucle principale. Le principe est clairement expliqué dans [127]. On peut ainsi casser assez facilement un DES à 8 ou à 10 tours. Le DES complet à 16 tours est resté hors de portée de cette attaque.

Cryptanalyse linéaire

La cryptanalyse linéaire a été introduite par H. Gilbert et par M. Matsui [77] dans le cas du DES. C'est une attaque à messages clairs connus. Elle n'est aussi utilisable que pour un DES restreint à quelques tours. Le DES réel n'est pas menacé par cette attaque.

Elle utilise de légers défauts statistiques des étages de substitutions (les boîtes S dans le cas du DES). En effet, la somme (XOR) de certaines entrées et de certaines sorties peut présenter un biais (des fréquences de 1 et de 0 légèrement différentes). On peut alors étudier la façon dont ces biais se succèdent avec l'empilement des étages et en déduire un biais global faisant intervenir certains bits du clair et certains bits obtenus au dernier étage (juste avant la dernière substitution), en remarquant que les valeurs des sous-clés rencontrées ne modifient pas ce biais (sauf son signe).

Les bits concernés au dernier étage peuvent être calculés en fonction du chiffré (pris parmi les couples clair/chiffré connus) et des bits correspondants de la dernière sous-clé (examinés par recherche exhaustive). Si les bits de la dernière sous-clé choisie correspondent à ceux de la clé réellement utilisée, le biais observé sur l'ensemble des couples clair/chiffré connus correspond probablement au biais théorique.

Cela permet de retrouver quelques bits de la dernière sous-clé. En examinant d'autres successions de biais dans l'empilement des étages, on peut parfois obtenir une partie importante de la dernière sous-clé et donc de la clé. Le reste peut alors être accessible à une recherche exhaustive.

Recherche exhaustive

L'accroissement de la puissance des ordinateurs, de leur nombre et des facilités de communication entre eux ont rendu envisageable une attaque de DES par recherche exhaustive de la clé (*brute force attack*). Pour démontrer que la taille des clés DES (56 bits) devenait insuffisante en regard des ressources de calcul maintenant disponibles, les laboratoires RSA ont lancé en Janvier 97 un défi consistant à décrypter par recherche exhaustive un message chiffré par DES. Une équipe coordonnant des milliers d'ordinateurs du monde entier a abouti à la découverte de la bonne clé le 17 Juin 1997, après avoir exploré environ un quart de l'espace des clés. Bien qu'une telle recherche demande des moyens considérables, elle a montré que le DES n'offre plus aujourd'hui une grande sécurité.

4. Triple-DES

Puisque la faiblesse de DES est la faible longueur de sa clé, il est naturel de chercher à combiner plusieurs chiffrements DES pour obtenir un système de chiffrement global avec une clé plus longue.

La première idée qui vient à l'esprit est de composer deux chiffrements DES avec des clés différentes. Mais on peut alors monter contre ce "double-DES" une attaque à messages clairs dite "par le milieu" parce qu'elle s'appuie sur le message intermédiaire (inconnu) apparaissant entre les deux chiffrements DES successifs. Cette attaque consiste à construire la liste des messages intermédiaires possibles en chiffrant par DES un clair avec les 2^{56} clés possibles. En déchiffrant par DES le chiffré correspondant avec des clés différentes, on obtient une autre liste de messages intermédiaires possibles et le véritable message intermédiaire est dans l'intersection des deux listes. Le coût en mémoire de cette attaque est très important mais son coût en temps n'est pas significativement plus élevé que l'attaque exhaustive sur DES.

Triple-DES consiste à composer deux chiffrements DES de même clé séparés par un déchiffrement DES avec une autre clé. Plus précisément :

$$\text{Triple-DES}_{k_1,k_2} = \text{DES}_{k_1} \circ \text{DES}_{k_2}^{-1} \circ \text{DES}_{k_1} .$$

Cette façon de faire est préférée à trois chiffrements parce qu'elle généralise DES qui se trouve être le cas particulier où $k_1 = k_2$. Bien sûr, le déchiffrement est

$$\text{Triple-DES}_{k_1,k_2}^{-1} = \text{DES}_{k_1}^{-1} \circ \text{DES}_{k_2} \circ \text{DES}_{k_1}^{-1} .$$

Une clé Triple-DES est donc composée de deux clés DES et fait 112 bits ce qui met Triple-DES largement hors de portée d'une attaque exhaustive. On peut aussi concevoir une variante à trois clés DES différentes mais elle reste vulnérable à une attaque de coût en 2^{112} s'appuyant sur l'un des deux messages intermédiaires.

5. Rijndael

Bien que la sécurité de Triple-DES semble élevée, il est un peu lent. Le besoin s'est fait sentir de définir un successeur à DES, entièrement nouveau, répondant à des critères modernes (par exemple en taille de clé), et susceptible de subvenir aux besoins cryptographiques pendant de nombreuses années. Un appel d'offres à été lancé par le NIST (National Institute of Standards and Technology) pour ce projet. Une quinzaine de candidats se sont mis sur les rangs. Cinq ont été retenus au cours de l'été 1999 (Mars, RC6, Rijndael, Serpent, TwoFish). C'est finalement Rijndael qui a été choisi en 2000 comme successeur du DES. On trouve une description complète de Rijndael dans [38].

C'est une création belge due à Joan Daemen et Vincent Rijmen. Comme l'imposait le cahier de charges de l'AES, il est spécifié dans plusieurs variantes utilisant des clés de tailles différentes : 128, 192 et 256 bits. De plus, il peut chiffrer des blocs de 128, 192 ou 256 bits. Les choix de la taille de la clé et de la taille des blocs sont indépendants, il y a donc en tout 9 variantes.

Description succincte

Pour faciliter la description de Rijndael, il faut disposer les octets des blocs selon une matrice $4 \times N_b$ et les octets de la clé selon une matrice $4 \times N_k$, où N_b et N_k valent 4, 6 ou 8 selon les longueurs respectives des blocs et de la clé.

Le chiffrement par Rijndael est une succession de tours semblables. Le nombre N_r de tours dépend de N_b et N_k. Il vaut 10 si $N_b = N_k = 4$, 12 si $\max(N_b, N_k) = 6$, et 14 si $\max(N_b, N_k) = 8$. Chaque tour est constitué de quatre phases : une substitution d'octets, un décalage par lignes, une opération sur chaque colonne, et une addition de clé partielle, sauf le dernier tour pour lequel l'opération sur chaque colonne est supprimée. Le premier tour est précédé d'une addition de clé partielle.

La substitution d'octets s'applique à chaque octet de la matrice du bloc que l'on chiffre. Chaque octet $a_7 \cdots a_0$ peut être interprété comme un élément $a_7 x^7 + \cdots + a_0$ du corps $\mathbb{F}_{256}$, défini comme extension du corps $\mathbb{F}_2$ au moyen du polynôme irréductible $x^8+x^4+x^3+x+1$ (c'est le premier pour l'ordre lexicographique). La substitution consiste à prendre l'inverse $b_7 x^7 + \cdots + b_0$ de l'octet considéré dans le corps $\mathbb{F}_{256}$ (en laissant l'octet inchangé s'il est nul) et à appliquer la transformation suivante :

$$\begin{pmatrix} c_0 \\ c_1 \\ c_2 \\ c_3 \\ c_4 \\ c_5 \\ c_6 \\ c_7 \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 1 & 1 & 0 & 0 & 0 & 1 & 1 & 1 \\ 1 & 1 & 1 & 0 & 0 & 0 & 1 & 1 \\ 1 & 1 & 1 & 1 & 0 & 0 & 0 & 1 \\ 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 \end{pmatrix} \begin{pmatrix} b_0 \\ b_1 \\ b_2 \\ b_3 \\ b_4 \\ b_5 \\ b_6 \\ b_7 \end{pmatrix} + \begin{pmatrix} 1 \\ 1 \\ 0 \\ 0 \\ 0 \\ 1 \\ 1 \\ 0 \end{pmatrix}.$$

Le résultat de la substitution est l'octet $c_7 \cdots c_0$.

Le décalage de lignes consiste à décaler cycliquement par la gauche la deuxième ligne d'un cran, la troisième ligne de deux crans (trois si $N_b = 8$), et la quatrième ligne de trois crans (quatre si $N_b = 8$). La première ligne reste inchangée.

L'opération sur les colonnes est définie en considérant chaque colonne comme un polynôme de degré < 4 sur $\mathbb{F}_{256}$ (le terme constant est donné par l'octet en haut de la colonne). On multiplie alors le polynôme associé à cette colonne par le polynôme $(1 + x)T^3 + T^2 + T + x$ modulo $T^4 - 1$. Le résultat de l'opération est la colonne associée à ce nouveau polynôme.

L'addition de clé partielle est un XOR bit à bit. Les $N_r + 1$ clés partielles sont dérivées de la clé et leur longueur est $4N_b$ octets. Il y a donc au total $4N_b(N_r + 1)$ octets de clés partielles.

Dérivation des clés partielles

Là encore, il est pratique de disposer les octets des clés partielles en une matrice constituée de $N_b(N_r + 1)$ colonnes de quatre octets chacune. Les N_k premières colonnes sont remplies par les octets de la clé. Pour calculer les colonnes suivantes, on prend les octets de la dernière colonne remplie, on les permute, et on applique la substitution décrite précédemment à chacun d'eux. On ajoute (XOR) au résultat la première colonne de la matrice de clé, et une constante, et cela forme la $N_k + 1$-ième colonne. Les colonnes de rang $N_k + 2$ à $2N_k$ sont obtenues de proche en proche en ajoutant (XOR) les colonnes 2 à N_k à la colonne précédente. On obtient les colonnes suivantes en ajoutant successivement la deuxième, puis la troisième, ainsi de suite jusqu'à la N_k-ième. Les blocs de N_k colonnes suivants sont obtenus de manière semblable.

6. Modes opératoires

Un message réel est en général composé de nombreux blocs. La façon la plus immédiate pour chiffrer un tel message est de chiffrer successivement chaque bloc, avec la même clé. Toutefois cette méthode, dite ECB (*Electronic Code-Book*), présente des inconvénients. En particulier, lorsque deux blocs du message (ou de deux messages) clair sont identiques, cela se voit sur le chiffré. De plus un attaquant actif peut permuter des blocs chiffrés et/ou en supprimer de telle façon que le clair modifié ait encore un sens, différent du sens

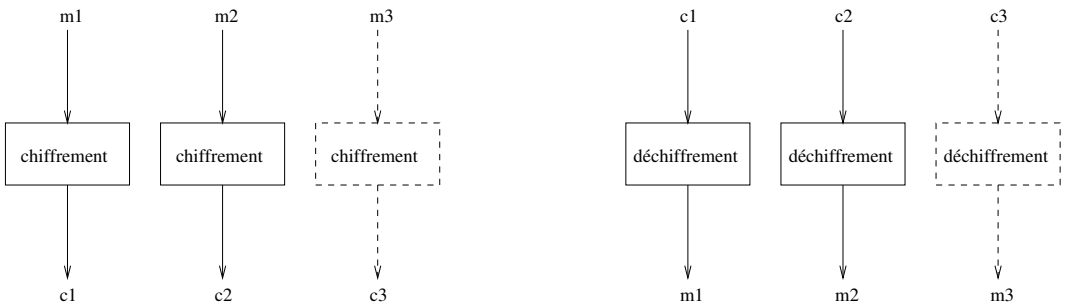

Figure 8. Le mode ECB

initial. La méthode ECB à toutefois l'avantage d'être parallélisable, de laisser la liberté de chiffrer/déchiffrer les blocs dans n'importe quel ordre et, en cas de perte d'un bloc chiffré, de ne pas bloquer le déchiffrement des blocs restants.

Pour palier à ces inconvénients, on utilise souvent des modes opératoires permettant de chaîner les blocs. Ainsi le mode CBC (*Cipher Bloc Chaining*) consiste à, avant le chiffrement d'un bloc, le masquer par le résultat du chiffrement du bloc précédent au moyen de l'opération XOR. Le premier bloc clair est lui aussi masqué, par une valeur habituellement notée IV (*Initial Value*) et de préférence variable (la date et l'heure peuvent faire une bonne IV) pour que deux chiffrements successifs du même message soient différents. La valeur initiale IV n'a pas besoin d'être secrète, et elle est en général transmise en clair avant le message chiffré. Noter que si le destinataire reçoit un bloc chiffré avec des bits erronés, cela affecte le déchiffrement de ce bloc et du suivant mais pas des autres.

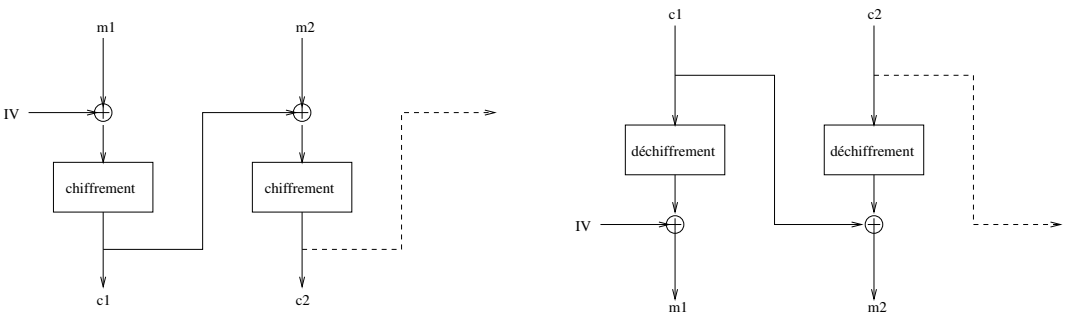

Figure 9. Le mode CBC

Pour déchiffrer un message en mode ECB ou CBC, il faut avoir reçu chaque bloc chiffré complet avant de pouvoir obtenir n'importe quel bit du bloc clair correspondant. Cependant, certaines applications nécessitent de pouvoir déchiffrer le flux des données au fur et à mesure de leur arrivée, même si la transmission se fait par petits morceaux. Cela est possible en utilisant le mode OFB (*Output FeedBack*). Celui-ci consiste à itérer la fonction de chiffrement sur une valeur initiale IV et à utiliser le flot de bits pseudo-aléatoires obtenus pour masquer les bits clairs à l'aide de l'opération XOR. Il est cette fois-ci très important que la valeur initiale IV soit différente pour chaque nouveau message. A noter que l'opération de déchiffrement est identique à celle de chiffrement, et utilise la fonction de chiffrement de blocs et non celle de déchiffrement. Remarquons un

inconvénient de cette méthode : un attaquant actif peut modifier des bits du message clair en modifiant les bits correspondants du chiffré (cela peut être aussi un avantage, si un bit du message chiffré est modifié par erreur au cours de la transmission, seul le bit correspondant du message clair sera affecté).

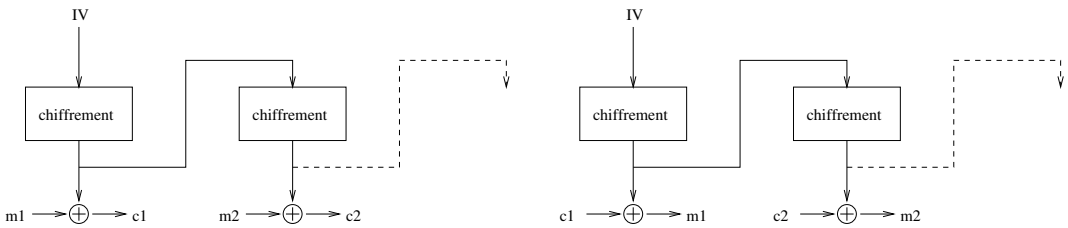

Figure 10. Le mode OFB

Enfin, le mode CFB (*Cipher FeedBack*) est proche du mode OFB mais le flot de bits pseudo-aléatoires dépend cette fois des blocs chiffrés. Un attaquant peut encore modifier un bit du clair en modifiant le bit correspondant du chiffré, mais alors le bloc suivant une fois déchiffré sera complètement différent du bloc original.

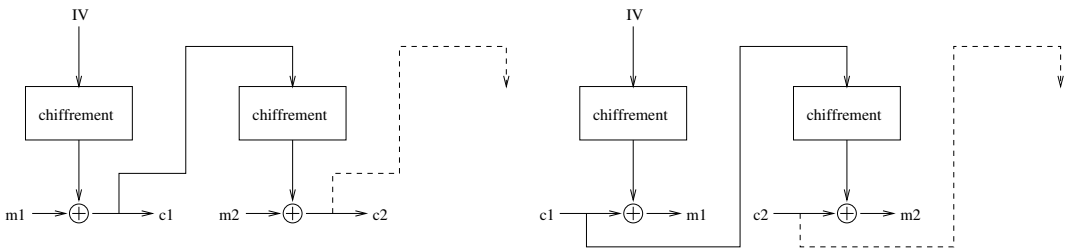

Figure 11. Le mode CFB

7. Sécurité des schémas de Feistel

Le monde de la cryptographie symétrique ne se prête pas facilement à des analyses rigoureuses de sécurité. Toutefois, des efforts ont été fournis dans ce sens et ont permis d'obtenir quelques résultats importants. Parmi eux, celui présenté ici donne une justification théorique à l'utilisation de schémas de Feistel. Ce résultat est dû à Luby & Rackoff [73] mais la démonstration bien plus simple indiquée ici est due à Maurer [79].

Dans un schéma de Feistel, les blocs sont coupés en deux et chaque moitié est modifiée à tour de rôle en lui appliquant une opération XOR avec une valeur dépendant de l'autre moitié. Cette valeur est calculée à l'aide d'une fonction F_k (au tour k) différente à chaque tour (car elle dépend en général de sous-clés différentes).

Un schéma de Feistel définit donc une fonction (même une permutation), qui à un bloc clair associe un bloc chiffré, dépendant des F_k. Toute fonction ne peut probablement pas être décrite par un schéma de Feistel (en tout cas avec un nombre de tours petit). Aussi, on peut se poser la question de savoir s'il existe un moyen algorithmique permettant de déterminer, rien qu'on observant l'action d'une fonction de chiffrement sur certains clairs, si cette fonction est construite au moyen d'un schéma de Feistel.

Lorsqu'on se restreint à des schémas de Feistel à deux tours, la réponse est *oui* et il suffit d'observer le chiffrement de deux blocs pour reconnaître presque à coup sûr un schéma de Feistel d'une fonction de chiffrement quelconque. En effet, il suffit de choisir deux blocs clairs $L_1 \| R$ et $L_2 \| R$ ayant la même partie droite R. Les chiffrés obtenus, dans le cas d'un schéma de Feistel à deux tours, ont pour partie gauche $L'_i = L_i \oplus F_1(R)$ et leur différence $L'_1 \oplus L'_2$ est égale à $L_1 \oplus L_2$. Au contraire, cette égalité a très peu de chance d'avoir lieu si la fonction de chiffrement n'est pas un schéma de Feistel. Cela donne donc un critère efficace pour reconnaître les schémas de Feistel.

Le résultat 7.1 de Luby & Rackoff montrera que, dès que le nombre de tours atteint 3, un tel algorithme n'existe pas, si on le contraint à examiner suffisamment peu de couples (clair/chiffré).

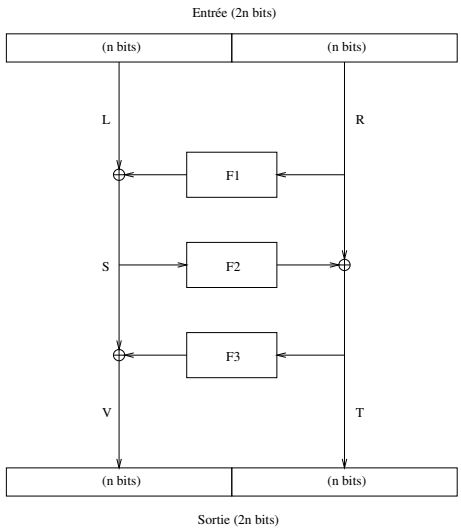

Figure 12. Construction de Luby & Rackoff : $\Phi(F_1, F_2, F_3)$

Pour $m \in \mathbb{N}$, notons $\mathcal{F}_m$ l'ensemble des fonctions de $\{0,1\}^m$ dans lui-même. Fixons $n \in \mathbb{N}^*$. On s'intéresse à deux distributions de $\mathcal{F}_{2n}$. La première est la distribution uniforme U.

Pour la deuxième, on considère la construction de Feistel, qui à chaque triplet (F_1, F_2, F_3) d'éléments de $\mathcal{F}_n$ associe la permutation $\Phi(F_1, F_2, F_3)$ correspondante et représentée par la figure 12. La distribution de Luby-Rackoff est la distribution LR_n sur $\mathcal{F}_{2n}$ obtenue en choisissant F_1, F_2 et F_3 dans $\mathcal{F}_n$ selon la distribution uniforme et en construisant la permutation $\Phi(F_1, F_2, F_3)$.

Un algorithme D (pour *distingueur*) susceptible de reconnaître les schémas de Feistel à k étages pourrait utiliser l'oracle à tester comme sous programme, et devrait pouvoir s'insérer dans le protocole suivant.

$b \leftarrow_U \{0,1\}$
Si $b = 1$ alors
  $F_1 \leftarrow_U \mathcal{F}_n, \ldots, F_k \leftarrow_U \mathcal{F}_k$
  $f \leftarrow \Phi(F_1, \ldots, F_k)$
Sinon $f \leftarrow_U \mathcal{F}_{2n}$
$b' \leftarrow D(f)$
$b \stackrel{?}{=} b'$

Le distingueur serait efficace si la réponse b' qu'il fournit est égale à la valeur de b avec probabilité meilleure que $1/2$ (la notation $a \leftarrow_U B$ signifie ici que a est affecté avec une valeur prise dans l'ensemble B, cette valeur étant choisie en suivant la distribution uniforme).

En fait cette probabilité peut s'exprimer sous la forme

$$\begin{aligned}\operatorname{prob}\{b=b'\} &= \operatorname{prob}\{b=1\}\operatorname{prob}\{b'=1|b=1\} + \operatorname{prob}\{b=0\}\operatorname{prob}\{b'=0|b=0\}\\ &= \frac{1}{2}\operatorname{prob}_{\mathrm{LR}_n}\{D(f)=1\} + \frac{1}{2}\operatorname{prob}_U\{D(f)=0\}\\ &= \frac{1}{2}\operatorname{prob}_{\mathrm{LR}_n}\{D(f)=1\} + \frac{1}{2}(1-\operatorname{prob}_U\{D(f)=1\})\\ &= \frac{1}{2} + \frac{1}{2}(\operatorname{prob}_{\mathrm{LR}_n}\{D(f)=1\} - \operatorname{prob}_U\{D(f)=1\})\end{aligned}$$

où $\operatorname{prob}_{\mathrm{LR}_n}$ signifie que f est choisie selon la distribution de Luby-Rackoff et prob_U signifie que f est choisie selon la distribution uniforme sur $\mathcal{F}_{2n}$. Donc les performances du distingueur D pourront être mesurées par la quantité $\operatorname{prob}_{\mathrm{LR}_n}\{D(f)=1\} - \operatorname{prob}_U\{D(f)=1\}$.

7.1. — Théorème (Luby-Rackoff). *Soit D un algorithme faisant au plus q appels distincts à un oracle $f \in \mathcal{F}_{2n}$. Alors*

$$\left|\operatorname{prob}_{\mathrm{LR}_n}\{D(f)=1\} - \operatorname{prob}_U\{D(f)=1\}\right| \leqslant q^2/2^n$$

où $\operatorname{prob}_{\mathrm{LR}_n}$ signifie que f est choisie selon la distribution de Luby-Rackoff et prob_U signifie que f est choisie selon la distribution uniforme sur $\mathcal{F}_{2n}$.

Démonstration — Lorsque $f = \Phi(F_1, F_2, F_3)$, on note respectivement L_i et R_i (pour $1 \leqslant i \leqslant q$) les parties gauches et droites des données soumises à l'oracle lors du i-ième appel, ainsi que $S_i = L_i \oplus F_1(R_i)$, $T_i = R_i \oplus F_2(S_i)$ et $V_i = S_i \oplus F_3(T_i)$ de telle sorte que $f(L_i \,\|\, R_i) = V_i \,\|\, T_i$. Si les S_i sont tous distincts, alors les $F_2(S_i)$ sont indépendants (car F_2 est uniformément aléatoire), donc les T_i sont uniformément aléatoires. De même, si les T_i sont tous distincts, alors les $F_3(T_i)$ sont indépendants et les V_i sont uniformément aléatoires. Ainsi, les réponses $V_i \,\|\, T_i$ de l'oracle $f = \Phi(F_1, F_2, F_3)$ sont indiscernables de celles que ferait un oracle choisi uniformément dans $\mathcal{F}_{2n}$, lorsque les S_i sont distincts et les T_i aussi.

La faculté de D à discerner les deux distributions ne peut donc s'exprimer que lorsque plusieurs S_i ou plusieurs T_i coïncident. Or, pour $i \neq j$, la probabilité que $S_i = S_j$ est $1/2^n$ lorsque $R_i \neq R_j$, puisque F_1 est uniformément aléatoire. Cette même probabilité est nulle lorsque $R_i = R_j$ car les appels $L_i \,\|\, R_i$ sont distincts. Dans tous les cas, la probabilité que $S_i = S_j$ est majorée par $1/2^n$. De même, la probabilité que $T_i = T_j$ est majorée par $1/2^n$. Au total, la probabilité que deux S_i ou deux T_i coïncident est majorée par

$$\left(\frac{q(q-1)}{2} + \frac{q(q-1)}{2}\right) \cdot 2^{-n} < q^2/2^n.$$

A fortiori, on a l'inégalité annoncée pour la faculté de D à discerner les deux distributions. □

Ce résultat montre que, pour distinguer avec un avantage non négligeable les schémas de Feistel à 3 étages parmi toutes les fonctions possibles, en menant une attaque à messages clairs choisis, le nombre de couples (clair/chiffré) à obtenir est au moins de l'ordre de $2^{n/2}$. Il se trouve que cette borne ne peut pas être significativement améliorée, comme cela est montré dans [7].

D'autre part, sous une attaque à messages clairs choisis **et** chiffrés choisis il est très facile de distinguer un schéma de Feistel à 3 étages. Pour cela, il suffit de soumettre en chiffrement deux requêtes de la forme $L_1 \,\|\, R$ et $L_2 \,\|\, R$, puis, en notant les résultats respectifs $V_1 \,\|\, T_1$ et $V_2 \,\|\, T_2$, de soumettre en déchiffrement la requête $V_2 \oplus L_1 \oplus L_2 \,\|\, T_2$. Notons alors $L' \,\|\, R'$ le clair obtenu. Un tel schéma de Feistel est alors trahi par la relation $R \oplus R' = T_1 \oplus T_2$. Il faut alors rajouter un quatrième étage pour obtenir la sécurité face à cette attaque.

Chapitre 3

Introduction à la Théorie de l'Information

1. Rappels de Théorie des Probabilités

Nous ne considérons ici que des espaces probabilisés *finis*.

Espaces probabilisés

1.1. — Définition. Un *espace probabilisé (fini)* et un triplet (S, A, p) possédant les propriétés suivantes :
• S est un ensemble fini, dont les éléments sont appelés *épreuves*.
• A est un ensemble de parties de S tel que

$$S \in A,$$
$$\overline{E} = S \setminus E \in A, \quad \forall E \in A,$$
$$E_1 \cup E_2 \in A, \quad \forall E_1, E_2 \in A.$$

Les éléments de A sont appelés *événements*.
• p est une application de A dans $[0, 1]$ telle que

$$p(\emptyset) = 0, \qquad p(S) = 1,$$
$$p(E_1 \cup E_2) = p(E_1) + p(E_2), \quad \forall E_1, E_2 \in A \text{ disjoints.}$$

Pour $E \in A$, $p(E)$ est appelé la *probabilité de* E.

Bien souvent, A est l'ensemble $\mathcal{P}(S)$ de toutes les parties de S. D'autre part, par passage au complémentaire, on a aussi la propriété suivante

$$E_1 \cap E_2 \in A, \quad \forall E_1, E_2 \in A.$$

Enfin, pour $s \in S$, on pose $p(s) = p(\{s\})$.

1.2. — Exemple. L'espace probabilisé correspondant au lancer d'un dé non pipé est le suivant :

$$S = \{1, 2, 3, 4, 5, 6\}, \qquad A = \mathcal{P}(S), \qquad \forall E \in A, p(E) = |E|/6.$$

1.3. — Exemple. On peut représenter le lancer de deux dés indiscernables et non pipés par l'espace probabilisé suivant :

$$S = \{(i, j) | 1 \leqslant i, j \leqslant 6\}, \qquad A = \{E \in \mathcal{P}(S) | (i, j) \in E \Rightarrow (j, i) \in E\}, \qquad \forall E \in A, p(E) = |E|/36.$$

1.4. — Proposition. *Soit (S, A, p) un espace probabilisé. On a les propriété suivantes :*
• *Pour $E \in A$, on a $p(\overline{E}) = 1 - p(E)$.*
• *Pour $E_1, E_2 \in A$ tels que $E_1 \subseteq E_2$, on a $p(E_1) \leqslant p(E_2)$.*
• *Pour $E_1, E_2 \in A$, on a $p(E_1 \cup E_2) = p(E_1) + p(E_2) - p(E_1 \cap E_2)$.*

Démonstration — Facile. □

1.5. — **Définition.** Soient E_1 et E_2 deux événements tels que $p(E_2) > 0$. La *probabilité conditionnelle de E_1, étant donné E_2* est

$$p(E_1|E_2) = \frac{p(E_1 \cap E_2)}{p(E_2)}.$$

1.6. — **Définition.** Deux événements E_1 et E_2 sont dits *indépendants* si $p(E_1 \cap E_2) = p(E_1)p(E_2)$. Cela revient à dire que $p(E_1|E_2) = p(E_1)$.

1.7. — **Théorème (Bayes).** *Soit E_1 et E_2 tels que $p(E_1) > 0$ et $p(E_2) > 0$. Alors*

$$p(E_2)p(E_1|E_2) = p(E_1)p(E_2|E_1).$$

Démonstration — Triviale. □

Variables aléatoires

1.8. — **Définition.** Soit (S, A, p) un espace probabilisé. Une *variable aléatoire sur* (S, A, p) est une application X de S dans $\mathbb{R}$ telle que l'image réciproque de chaque réel λ par X soit un événement (i.e. un élément de A). On note cet événement $[X = \lambda]$ et on pose $p_X(\lambda) = p([X = \lambda])$.

1.9. — **Définition.** Soit X une variable aléatoire sur un espace probabilisé (S, A, p). L'*espérance de X* est le nombre réel

$$E(X) = \sum_{\lambda \in \mathbb{R}} p_X(\lambda)\lambda.$$

1.10. — **Définition.** Soit X une variable aléatoire sur un espace probabilisé. La *variance de X* est le nombre réel

$$\mathrm{Var}(X) = E\big((X - E(X))^2\big) = E(X^2) - E(X)^2.$$

1.11. — **Définition.** Deux variables aléatoires X et Y sur un même espace probabilisé sont dites *indépendantes* si, pour tous $\lambda, \mu \in \mathbb{R}$, les événements $[X = \lambda]$ et $[Y = \mu]$ sont indépendants.

1.12. — **Théorème.** *Soient X et Y deux variables aléatoires sur un même espace probabilisé et soit $a, b \in \mathbb{R}$. alors*

$$E(aX + bY) = aE(X) + bE(Y)$$

et, si X et Y sont indépendantes,

$$E(XY) = E(X)E(Y).$$

Démonstration — La première égalité est très facile. Montrons la deuxième :

$$\begin{aligned} E(XY) &= \sum_{\lambda \in \mathbb{R}} p([XY = \lambda])\,\lambda = \sum_{\lambda \in \mathbb{R}} \Big(\sum_{\mu\nu = \lambda} p([X = \mu] \cap [Y = \nu]) \Big)\,\mu\nu \\ &= \sum_{\mu,\nu \in \mathbb{R}} p([X = \mu] \cap [Y = \nu])\,\mu\nu = \sum_{\mu,\nu \in \mathbb{R}} p([X = \mu])p([Y = \nu])\,\mu\nu \\ &= \Big(\sum_{\mu \in \mathbb{R}} p([X = \mu])\,\mu \Big) \Big(\sum_{\nu \in \mathbb{R}} p([Y = \nu])\,\nu \Big) = E(X)E(Y). \end{aligned}$$

□

1.13. — **Définition.** Soient X, Y deux variables aléatoires sur un même espace probabilisé. La *covariance de X et Y* est le nombre réel

$$\mathrm{Covar}(X, Y) = E\big((X - E(X))(Y - E(Y))\big).$$

1.14. — **Théorème.** *Soient X et Y deux variables aléatoires sur un même espace probabilisé et soit $a, b \in \mathbb{R}$. alors*

$$\mathrm{Var}(aX + bY) = a^2\,\mathrm{Var}(X) + b^2\,\mathrm{Var}(Y) + 2ab\,\mathrm{Covar}(X, Y)$$

et, si X et Y sont indépendantes,

$$\mathrm{Var}(X+Y) = \mathrm{Var}(X) + \mathrm{Var}(Y).$$

DÉMONSTRATION — Posons $\mu = E(X)$ et $\nu = E(Y)$. On a donc $E(aX+bY) = a\mu + b\nu$ d'après 1.12. Donc,

$$\begin{aligned}\mathrm{Var}(aX+bY) &= E\big((aX+bY-(a\mu+b\nu))^2\big) = E\big((a(X-\mu)+b(Y-\nu))^2\big)\\ &= E\big(a^2(X-\mu)^2 + b^2(Y-\nu)^2 + 2ab(X-\mu)(Y-\nu)\big)\\ &= a^2E\big((X-\mu)^2\big) + b^2E\big((Y-\nu)^2\big) + 2abE\big((X-\mu)(Y-\nu)\big)\\ &= a^2\,\mathrm{Var}(X) + b^2\,\mathrm{Var}(Y) + 2ab\,\mathrm{Covar}(X,Y).\end{aligned}$$

Pour la deuxième égalité, il suffit de montrer que, si X et Y sont indépendantes, alors $\mathrm{Covar}(X,Y) = 0$. Mais on voit facilement que les variables aléatoires $X-\mu$ et $Y-\nu$ sont indépendantes. Donc,

$$\mathrm{Covar}(X,Y) = E\big((X-\mu)(Y-\nu)\big) = E\big((X-\mu)\big)E\big((Y-\nu)\big) = 0\cdot 0 = 0.$$

□

1.15. — **Exemple.** Dans l'espace probabilisé 1.2, on considère la variable aléatoire que à $s \in S$ associe $s \in \mathbb{R}$. L'espérance de cette variable aléatoire est $7/2$. Sa variance est $35/12$. Dans l'espace probabilisé 1.3, on considère la variable aléatoire qui à (i,j) associe $i+j$. Son espérance est 7. Sa variance est $35/6$.

1.16. — **Exercice.** On fait n lancer à pile ou face avec une pièce truquée pour laquelle la probabilité d'obtenir face est p. On s'intéresse à la variable aléatoire donnant le nombre de fois où on obtient face. Déterminer sa moyenne et sa variance. (Les réponses sont respectivement np et $np(1-p)$.)

2. Paradoxe des anniversaires

Dans une classe de 23 élèves, on demande à chacun d'eux la date de son anniversaire. Avec probabilité supérieure à $1/2$, au moins deux élèves ont leur anniversaire le même jour. Cela peut paraître surprenant, si on ne réfléchit pas trop, puisqu'il y a 365 jours dans l'année. D'où le terme de paradoxe, souvent utilisé.

Formellement, le problème est le suivant. On tire au hasard suivant une distribution uniforme k valeurs parmi n. La probabilité qu'il n'y ait **pas** coïncidence est

$$1-p = (1-\frac{1}{n})(1-\frac{2}{n})\cdots(1-\frac{k-1}{n}) = \prod_{i=1}^{k-1}(1-\frac{i}{n}).$$

Or, on a l'inégalité $1-x \leqslant e^{-x}$ pour tout $x \in \mathbb{R}$. Donc,

$$1-p \leqslant \prod_{i=1}^{k-1} e^{-i/n} = e^{-k(k-1)/2n} \leqslant e^{-(k-1)^2/2n}.$$

Il y a donc au moins une coïncidence avec probabilité

$$p \geqslant 1 - e^{-k(k-1)/2n} \geqslant 1 - e^{-(k-1)^2/2n}.$$

Si on veut que p soit au moins égal à $p_0 \in [0,1]$, il suffit donc que k vérifie

$$k \geqslant \sqrt{-2n\ln(1-p_0)} + 1.$$

Pour avoir un bonne chance (e.g. $p_0 = 0.95$) d'obtenir une collision en tirant au hasard k valeurs dans un ensemble à n éléments, il suffit donc que k soit de l'ordre de $\sqrt{n}$ multiplié par un petit facteur constant ($-2\ln(0.05) < 6$, dans cet exemple).

En reprenant l'exemple des anniversaires, on a $n = 365$ et, pour $p_0 = 1/2$, la condition suffisante ci-dessus s'écrit $k \geqslant 24$. Si on utilise l'inégalité plus fine faisant apparaître $k(k-1)$ au lieu de $(k-1)^2$, on se rend compte que la condition $k \geqslant 23$ suffit.

3. Eléments de Théorie de l'Information

La Théorie de l'Information est due à Shannon [122]. Ses liens avec la cryptographie ont été développés par Hellman [51].

Considérons l'expérience consistant à lancer une pièce de monnaie et à relever le côté où elle tombe (pile ou face). Le résultat d'une telle expérience peut se coder par un chiffre binaire. On dira que l'*information* contenue dans ce résultat est de 1 bit.

Considérons maintenant un dé tétraédrique dont les quatre faces sont désignées par A, B, C et D. Le résultat de chaque lancer de ce dé peut se coder sur deux bits (par exemple, $A \to 00$, $B \to 01$, $C \to 10$ et $D \to 11$). Il faudra $2n$ bits pour représenter le résultat de n lancers.

Modifions le dé tétraédrique de telle manière que les probabilités d'obtenir les valeurs A, B, C et D soient inégales. Par exemple, supposons que ces probabilités sont respectivement $1/2$, $1/4$, $1/8$ et $1/8$. Considérons alors le codage suivant.

$$A \to 0, \qquad B \to 10, \qquad C \to 110, \qquad D \to 111.$$

On peut alors coder n'importe quelle séquence de résultats de lancers par une suite de bits. Par exemple la séquence $ADBABCA$ sera codée par 0111100101100. D'autre part, puisque aucun des quatre codes choisis n'est préfixe d'un autre, la donnée d'une séquence de bits obtenue de cette façon permet de retrouver sans ambiguïté la séquence de lancers initiale.

Combien faut-il de bits en moyenne pour représenter les résultats d'un lancer ? En tenant compte des probabilités de chaque résultat, on obtient $\frac{1}{2} \times 1 + \frac{1}{4} \times 2 + \frac{1}{8} \times 3 + \frac{1}{8} \times 3 = 7/4$. Cette valeur est strictement inférieure à 2. Il faut donc en moyenne moins de deux bits pour représenter le résultat d'un lancer.

Dans cette façon de coder le résultat d'un lancer, on a utilisé une suite de bits dont la longueur était proportionnelle à $-\log(p)$, où p est la probabilité de ce résultat. On peut montrer que cette façon de faire est optimale si l'on veut que le nombre moyen de bits par lancer soit minimum. Ce nombre moyen est l'*entropie* de l'expérience et représente l'information moyenne contenue dans un résultat.

Entropie

3.1. — Définition. Soit X une variable aléatoire discrète dont les issues possibles x_i ont chacune pour probabilité p_i $(1 \leqslant i \leqslant n)$. L'*entropie* de X est définie par

$$H(X) = -\sum_{i=1}^{n} p_i \log p_i.$$

La valeur numérique de l'entropie dépend linéairement de la base de logarithmes utilisée. Dans le cas de la base 2, on exprime l'entropie en bits, dans le cas de la base 10 en dits.

3.2. — Définition. Une fonction à valeurs réelles définie sur un intervalle I est dite *strictement concave* si on a l'inégalité

$$f\Big(\frac{x+y}{2}\Big) > \frac{f(x)+f(y)}{2}$$

pour tous $x, y \in I$ distincts.

3.3. — Théorème (inégalité de Jensen). *Soit f une fonction réelle continue strictement concave sur un intervalle I et soient $a_i > 0$ $(1 \leqslant i \leqslant n)$ des réels tels que $\sum_{i=1}^{n} a_i = 1$. Alors on a*

$$f\Big(\sum_{i=1}^{n} a_i x_i\Big) \geqslant \sum_{i=1}^{n} a_i f(x_i).$$

De plus, on a égalité si et seulement si les x_i sont tous égaux. □

3.4. — Théorème. *Soit X une variable aléatoire dont les n issues ont pour probabilités p_i, pour $1 \leqslant i \leqslant n$. On a*

$$H(X) \leqslant \log(n).$$

De plus, il y a égalité si et seulement si chaque p_i vaut $1/n$.

Démonstration — C'est une conséquence facile de l'inégalité 3.3 de Jensen. Poser $a_i = p_i$, $x_i = 1/p_i$ et $f = \log$. □

3.5. — Théorème. *Soient X, Y deux variables aléatoires. On a la relation*

$$H(X,Y) \leqslant H(X) + H(Y).$$

De plus, l'égalité est vérifiée si et seulement si X et Y sont indépendantes.

DÉMONSTRATION — Soient x_i $(1 \leqslant i \leqslant m)$ les différentes issues de X et p_i leurs probabilités respectives. De même, soient y_j $(1 \leqslant j \leqslant n)$ les différentes issues de Y et q_j leurs probabilités respectives. Enfin, pour $1 \leqslant i \leqslant m$ et $1 \leqslant j \leqslant n$, soit $r_{i,j}$ la probabilité de l'événement $\{X = x_i, Y = y_j\}$. On a donc

$$\sum_{j=1}^{n} r_{i,j} = p_i \quad \text{pour } 1 \leqslant i \leqslant m \qquad \text{et} \qquad \sum_{i=1}^{m} r_{i,j} = q_j \quad \text{pour } 1 \leqslant j \leqslant n.$$

Donc,

$$\begin{aligned} H(X) + H(Y) &= -\sum_{i=1}^{m} p_i \log p_i - \sum_{j=1}^{n} q_j \log q_j = -\sum_{i=1}^{m}\sum_{j=1}^{n} r_{i,j} \log p_i - \sum_{j=1}^{n}\sum_{i=1}^{m} r_{i,j} \log q_j \\ &= -\sum_{\substack{1\leqslant i\leqslant m \\ 1\leqslant j\leqslant n}} r_{i,j} \log(p_i q_j) \end{aligned}$$

tandis que

$$H(X,Y) = -\sum_{\substack{1\leqslant i\leqslant m \\ 1\leqslant j\leqslant n}} r_{i,j} \log r_{i,j}.$$

Ainsi,

$$\begin{aligned} H(X,Y) - H(X) - H(Y) &= -\sum_{\substack{1\leqslant i\leqslant m \\ 1\leqslant j\leqslant n}} r_{i,j} \log r_{i,j} + \sum_{\substack{1\leqslant i\leqslant m \\ 1\leqslant j\leqslant n}} r_{i,j} \log(p_i q_j) \\ &= \sum_{\substack{1\leqslant i\leqslant m \\ 1\leqslant j\leqslant n}} r_{i,j} \log(p_i q_j / r_{i,j}) \\ &\leqslant \log \sum_{\substack{1\leqslant i\leqslant m \\ 1\leqslant j\leqslant n}} p_i q_j \quad \text{par le théorème 3.3} \\ &= \log 1 = 0. \end{aligned}$$

D'où l'inégalité.

De plus, l'égalité est vérifiée si et seulement si toutes les valeurs $p_i q_j / r_{i,j}$ sont égales à une même constante c. Mais

$$\sum_{\substack{1\leqslant i\leqslant m \\ 1\leqslant j\leqslant n}} r_{i,j} = 1 \qquad \text{et} \qquad \sum_{\substack{1\leqslant i\leqslant m \\ 1\leqslant j\leqslant n}} p_i q_j = 1$$

donc $c = 1$. Cela signifie que $p_i q_j = r_{i,j}$ pour chaque couple (i,j) et donc que les variables X et Y sont indépendantes. □

3.6. — Définition. Soient X, Y deux variables aléatoires d'issues respectives x_i et y_j pour $1 \leqslant i \leqslant m$ et $1 \leqslant j \leqslant n$. On définit *l'entropie de X conditionnellement à Y*, notée $H(X|Y)$ par

$$\begin{aligned} H(X|Y) &= \sum_{j=1}^{n} p(y_j) H(X|y_j) \quad \text{où } H(X|y_j) = -\sum_{i=1}^{m} p(x_i|y_j) \log p(x_i|y_j) \\ &= -\sum_{\substack{1\leqslant i\leqslant m \\ 1\leqslant j\leqslant n}} p(x_i, y_j) \log p(x_i|y_j). \end{aligned}$$

3.7. — Théorème. *Soient X, Y deux variables aléatoires. On a*

$$H(X,Y) = H(X|Y) + H(Y).$$

Démonstration — On reprend les notations de la preuve précédente. On a donc $p(x_i|y_j) = r_{i,j}/q_j$ pour $1 \leqslant i \leqslant m$ et $1 \leqslant j \leqslant n$. Ainsi

$$\begin{aligned} H(X|Y) + H(Y) &= -\sum_{\substack{1\leqslant i\leqslant m\\ 1\leqslant j\leqslant n}} r_{i,j}\log(r_{i,j}/q_j) - \sum_{1\leqslant j\leqslant n} q_j \log q_j \\ &= -\sum_{\substack{1\leqslant i\leqslant m\\ 1\leqslant j\leqslant n}} r_{i,j}\log(r_{i,j}/q_j) - \sum_{\substack{1\leqslant i\leqslant m\\ 1\leqslant j\leqslant n}} r_{i,j}\log q_j \\ &= -\sum_{\substack{1\leqslant i\leqslant m\\ 1\leqslant j\leqslant n}} r_{i,j}\log r_{i,j} = H(X,Y) \end{aligned}$$

est l'égalité cherchée. □

Entropie d'un langage

Dans une suite aléatoire de lettre prises dans l'alphabet usuel, l'information portée par chacune des lettres est $\log_2(26) \simeq 4,7$ bits. En tenant compte des inégalités de fréquences des différentes lettres, on obtient $H(\mathbb{Z}_{26}) \simeq 4.19$ bits en Anglais et 4.14 bits en Français. On peut aussi tenir compte des fréquences de chaque digramme et on obtient alors $H(\mathbb{Z}_{26}^2) \simeq 7,8$ bits, soit seulement 3,9 bits par lettre. On peut en général tenir compte des fréquences des n-grammes, ce qui mène à la définition suivante.

3.8. — Définitions. Soit L un langage naturel sur un alphabet A. Son *entropie* est définie par

$$H(L) = \lim_{n\to\infty} \frac{H(A^n)}{n}.$$

La *redondance* de L est définie par

$$R(L) = 1 - \frac{H(L)}{\log|A|}.$$

Les études faites pour déterminer l'entropie de la langue anglaise ont mené à des valeurs de l'ordre de $1,25$ bits. En moyenne, une lettre d'un texte en Anglais ne porte donc guère plus d'un bit d'information. Ce genre de situation est exploitée par les algorithmes de compression de fichiers, qui sont particulièrement efficaces pour les fichiers texte. Dans un fichier compressé, l'information moyenne portée par chaque bit est proche de 1 ; le fichier compressé est donc beaucoup plus court, tout en portant globalement la même information que le fichier texte original.

3.9. — Exercice. • Quelle quantité d'information porte une plaque minéralogique, avec quatre chiffres, deux lettres et encore deux chiffres ?

• Quelle est la quantité d'information représentée par un tirage du loto (6 boules parmi 49) ?

3.10. — Exercice. On considère les deux expériences suivantes. Pour la première, on dispose d'une urne contenant 10 boules blanches, 5 noires et 5 rouges. On tire une boule de cette urne et on note sa couleur. La deuxième expérience est identique sauf que l'urne contient 8 boules blanches, 8 noires et 4 rouges. Quelle est l'expérience dont l'issue est la plus incertaine ?

3.11. — Exercice. On possède 25 pièces de monnaie d'apparence identique. Cependant, l'une d'elles est plus légère que les autres (qui elles, sont identiques). On dispose d'une balance à plateaux, sans poids auxiliaires. Combien faut-il de pesées pour déterminer quelle est la pièce la plus légère ? On déterminera d'abord une borne inférieure à ce nombre de pesées, puis on donnera un algorithme pour lequel le nombre de pesées ne dépasse pas cette borne.

4. Sécurité parfaite

Lors de l'étude des systèmes cryptographiques classiques, nous avons décrit comment on pouvait parfois décrypter certains messages, en utilisant les disparités existant dans le fréquences d'occurence des lettres de messages clairs. Nous allons nous intéresser à ce genre de situation en général.

Soit $(\mathcal{P}, \mathcal{C}, \mathcal{K})$ un système cryptographique, et supposons que $\mathcal{P}$ est probabilisé (nous noterons $p_{\mathcal{P}}$ la fonction de probabilités sur $\mathcal{P}$), de manière à tenir compte de ces disparités. Nous supposerons aussi que $\mathcal{K}$ est probabilisé, ce qui permet de tenir compte du fait que la méthode utilisée par Alice et Bob pour choisir une clé ne donne pas nécessairement chaque clé avec la même probabilité. On obtient alors une fonction de probabilité sur $\mathcal{C}$ définie par

$$p_{\mathcal{C}}(y) = \sum_k p_{\mathcal{K}}(k) p_{\mathcal{P}}(d_k(y)),$$

la somme étant indexée par toutes les clés k telles que y soit dans l'image de e_k. Pour chaque $y \in \mathcal{C}$, la probabilité $p_{\mathcal{C}}(y)$ décrit donc la probabilité d'obtenir le message chiffré y.

Voici un exemple d'un tel système cryptographique, avec $\mathcal{P} = \{a, b\}$, $\mathcal{C} = \{1, 2, 3, 4\}$ et $\mathcal{K} = \{k_1, k_2, k_3\}$. Les fonctions de chiffrement sont décrites par la table suivante. On y a indiqué entre parenthèses les valeurs de $p_{\mathcal{P}}$, celles de $p_{\mathcal{K}}$, et les produits correspondants.

	a (1/4)	b (3/4)
k_1 (1/2)	1 (1/8)	2 (3/8)
k_2 (1/4)	2 (1/16)	3 (3/16)
k_3 (1/4)	3 (1/16)	4 (3/16)

Figure 1. Exemple de système cryptographique probabilisé

En faisant la somme des valeurs indiquées pour chaque élément de $\mathcal{C}$, on obtient les valeurs de $p_{\mathcal{C}}$, à savoir $p_{\mathcal{C}}(1) = 1/8$, $p_{\mathcal{C}}(2) = 7/16$, $p_{\mathcal{C}}(3) = 1/4$ et $p_{\mathcal{C}}(4) = 3/16$. On remarque que $p_{\mathcal{P}}(a|3) = p_{\mathcal{P}}(a)$ et que $p_{\mathcal{P}}(b|3) = p_{\mathcal{P}}(b)$. Cela signifie que le fait que le message chiffré soit 3 n'apporte aucune information sur le message clair. Ce n'est pas vrai pour les autres valeurs de $\mathcal{C}$.

4.1. — Définition. *Un système cryptographique $(\mathcal{P}, \mathcal{C}, \mathcal{K})$ vérifie la propriété de sécurité parfaite si, pour tous $x \in \mathcal{P}$ et $y \in \mathcal{C}$, on a $p_{\mathcal{P}}(x|y) = p_{\mathcal{P}}(x)$*

Cela signifie que, pour chaque $x \in \mathcal{P}$ et $y \in \mathcal{C}$, les événements associés à x et à y sont indépendants. On peut aussi écrire la condition de sécurité parfaite sous la forme

$$H(\mathcal{P}|\mathcal{C}) = H(\mathcal{P}).$$

4.2. — Exercice. Calculer les entropies $H(\mathcal{P})$, $H(\mathcal{K})$ et $H(\mathcal{C})$ dans le cas du système cryptographique de la figure 1.

4.3. — Lemme. *Soit $(\mathcal{P}, \mathcal{C}, \mathcal{K})$ un système cryptographique vérifiant la propriété de sécurité parfaite et tel que $p_{\mathcal{C}}(y) > 0$ pour tout $y \in \mathcal{C}$. On a les inégalités $|\mathcal{K}| \geqslant |\mathcal{C}| \geqslant |\mathcal{P}|$.*

Démonstration — Fixons $x \in \mathcal{P}$ tel que $p_{\mathcal{P}}(x) > 0$. Pour chaque $y \in \mathcal{C}$, on a $p_{\mathcal{P}}(x|y) = p_{\mathcal{P}}(x)$. D'après le théorème 1.7 de Bayes, on obtient $p_{\mathcal{C}}(y|x) = p_{\mathcal{C}}(y) > 0$. Cela signifie qu'il existe au moins une clé k vérifiant $e_k(x) = y$. On a donc $|\mathcal{K}| \geqslant |\mathcal{C}|$. D'autre part, l'inégalité $|\mathcal{C}| \geqslant |\mathcal{P}|$ est toujours vérifiée puisque les fonctions e_k sont injectives. □

4.4. — Théorème. *Soit un système cryptographique $(\mathcal{P}, \mathcal{C}, \mathcal{K})$ vérifiant $|\mathcal{P}| = |\mathcal{C}| = |\mathcal{K}|$ ainsi que $p_{\mathcal{C}}(y) > 0$ pour tout $y \in \mathcal{C}$. Alors, il est à sécurité parfaite si et seulement si toutes les clés sont équiprobables et que, pour chaque $x \in \mathcal{P}$ et $y \in \mathcal{C}$, il existe une unique clé k vérifiant $e_k(x) = y$.*

DÉMONSTRATION — Supposons les conditions vérifiées. Alors, pour chaque $y \in \mathcal{C}$, on a

$$\begin{aligned} p_{\mathcal{C}}(y) &= \sum_{k|y\in \operatorname{Im} e_k} p_{\mathcal{K}}(k) p_{\mathcal{P}}(d_k(y)) \\ &= \frac{1}{|K|} \sum_{k\in\mathcal{K}} p_{\mathcal{P}}(d_k(y)) \qquad \text{car pour tout } k \in \mathcal{K},\ p_{\mathcal{K}}(k) = 1/|K| \text{ et } \operatorname{Im} e_k = \mathcal{C}. \\ &= \frac{1}{|K|} \sum_{x\in\mathcal{P}} p_{\mathcal{P}}(x) = \frac{1}{|K|}. \end{aligned}$$

D'autre part, pour $x \in \mathcal{P}$ et $y \in \mathcal{C}$,

$$p_{\mathcal{C}}(y|x) = \sum_{k|e_k(x)=y} p_{\mathcal{K}}(k) = \frac{1}{|K|}$$

puisque exactement une clé vérifie $e_k(x) = y$. En appliquant le théorème de Bayes,

$$p_{\mathcal{P}}(x|y) = \frac{p_{\mathcal{P}}(x) p_{\mathcal{C}}(y|x)}{p_{\mathcal{C}}(y)} = \frac{p_{\mathcal{P}}(x)/|K|}{1/|K|} = p_{\mathcal{P}}(x).$$

La sécurité du système cryptographique est donc parfaite.

Inversement, supposons que la sécurité est parfaite. Comme dans le lemme précédent, on montre que pour chaque couple $(x, y) \in \mathcal{P} \times \mathcal{C}$, il existe une clé k telle que $e_k(x) = y$. Pour x fixé, l'ensemble des $e_k(x)$ est donc de cardinal $|\mathcal{C}| = |\mathcal{K}|$. Ces $e_k(x)$ sont donc distincts deux-à-deux et, pour chaque $y \in \mathcal{C}$, la clé vérifiant $e_k(x) = y$ est unique.

Pour $k_1, k_2 \in \mathcal{K}$, comparons $p_{\mathcal{K}}(k_1)$ et $p_{\mathcal{K}}(k_2)$. Fixons $y \in \mathcal{C}$ et désignons x_1, x_2 les uniques éléments de $\mathcal{P}$ vérifiant $e_{k_1}(x_1) = y$ et $e_{k_2}(x_2) = y$. Par le théorème de Bayes, l'hypothèse de sécurité parfaite donne

$$p(y) = p(y|x_1) = \sum_{k|e_k(x_1)=y} p_{\mathcal{K}}(k) = p_{\mathcal{K}}(k_1).$$

De la même façon, on obtient $p(y) = p_{\mathcal{K}}(k_2)$. Les clés sont donc équiprobables. □

4.5. — **Exemple.** Le "One Time Pad" est à sécurité parfaite (lorsque les clés sont équiprobables, mais quelle que soit la distribution de probabilités sur $\mathcal{P}$).

Chapitre 4

Fonctions de hachage

En informatique traditionnelle, une fonction de hachage (traduction discutable de *hash function*) est un algorithme simple aidant à gérer certaines bases de données. La place où est inséré un article dans une telle base est déterminée par la valeur calculée par cette fonction. Cette valeur dépend en général de la totalité de l'article afin que le remplissage de la base de donnée se répartisse de façon équilibrée même si de nombreux articles se ressemblent.

La notion de *fonction de hachage* en cryptographie est plus restrictive. C'est une fonction d'un ensemble infini $\mathcal{M}$ (l'espace des messages) dans un ensemble fini $\mathcal{E}$ (l'espace des empreintes), facilement calculable, et possédant une ou plusieurs des propriétés décrites ci-après. En général, l'espace des messages est l'ensemble $\{0,1\}^*$ des mots binaires et l'espace des empreintes celui des trains de longueur fixée, comme $\{0,1\}^{128}$ ou $\{0,1\}^{160}$.

0.1. — Définitions. • Une fonction de hachage h est dite *à sens unique* si, pour essentiellement toutes les valeurs y de l'espace des empreintes, il est difficile de trouver un message x tel que $h(x) = y$.

• Une fonction de hachage h est dite *faiblement résistante aux collisions* si, pour essentiellement chaque message x, il est difficile de trouver un message $x' \neq x$ ayant même empreinte $h(x) = h(x')$.

• Une fonction de hachage h est dite *(fortement) résistante aux collisions* s'il est difficile de trouver deux messages x et x' ayant même empreinte $h(x) = h(x')$.

La propriété de résistance forte aux collisions implique clairement la propriété de résistance faible. Si on suppose en plus que les images réciproques ne sont jamais des singletons, elle implique aussi la propriété de sens unique.

Le "paradoxe des anniversaires" (section 3.2) montre qu'une recherche de collision $h(x) = h(x')$, où ni x ni x' ne sont imposés, est seulement de complexité $O(\sqrt{m})$, si m est le cardinal de l'ensemble des valeurs de h. Il existe donc des fonctions de hachage qui sont faiblement résistantes aux collisions mais pour lesquelles la propriété de résistance forte n'est pas vérifiée.

Une fonction de hachage permet d'obtenir à partir d'une donnée de longueur quelconque, une *empreinte* de taille réduite et fixe, mais caractéristique de la donnée (penser aux empreintes digitales), et pratiquement impossible à reproduire à partir d'une donnée différente. Les fonctions de hachage jouent donc un rôle important en authentification et signature. Mais, de plus en plus, elles trouvent aussi des applications dans les autres domaines de la cryptographie moderne. (Bien que leur construction, pour des raisons de rapidité, fasse en général intervenir des techniques semblables à celles de la cryptographie symétrique.)

1. Construction de fonctions de hachage

De nombreuses fonctions de hachage sont construites à partir d'une *fonction de compression* f (une fonction de $\{0,1\}^n$ dans $\{0,1\}^m$ avec $m < n$) selon le schéma suivant. Le message x est complété par un procédé réversible afin que sa longueur soit un multiple de $n - m$ (par exemple en ajoutant un 1 puis un nombre adéquat de 0), puis il est découpé en blocs x_i ($1 \leqslant i \leqslant t$) de longueur $n - m$. L'empreinte $h(x)$ est alors calculée selon les formules

$$H_0 = \mathrm{IV}, \qquad H_i = f(x_i \,\|\, H_{i-1}), \quad \text{pour } 1 \leqslant i \leqslant t, \qquad h(x) = H_t$$

où IV (*Initial Value*) désigne une constante dans $\{0,1\}^m$.

On peut imaginer d'utiliser une fonction de chiffrement par blocs pour construire une fonction de compression utilisable dans le schéma ci-dessus. Sont adaptées les fonctions de chiffrement pour lesquelles la longueur des blocs est égale à celle des clés et assez grande (par exemple 160) pour que la recherche de

collisions soit difficile. Un certain nombre de telles constructions se sont révélées non sûres, mais les quatre suivantes semblent être sûres.

$$f(x \,\|\, k) = e_k(x) \oplus x$$
$$f(x \,\|\, k) = e_k(x) \oplus x \oplus k$$
$$f(x \,\|\, k) = e_k(x \oplus k) \oplus x$$
$$f(x \,\|\, k) = e_k(x \oplus k) \oplus x \oplus k$$

En modifiant légèrement ce schéma, on obtient avec certitude une fonction de hachage résistante aux collisions pour peu que la fonction de compression le soit :

1.1. — Théorème (Damgård–Merkle). *Soit f une fonction de compression de $\{0,1\}^n$ dans $\{0,1\}^m$ avec $n > m+1$ résistante aux collisions. Si les messages sont complétés en un nombre entiers de blocs de longueur $n-m-1$ par un procédé réversible alors on obtient une fonction de hachage résistante aux collisions en posant*

$$H_1 = f(0^{m+1} \,\|\, x_1), \qquad H_i = f(H_{i-1} \,\|\, 1 \,\|\, x_{i-1}) \quad \textit{pour } 2 \leqslant i \leqslant t.$$

DÉMONSTRATION — Voir [127], page 242. □

2. Fonctions de hachage classiques

SHA-1

Elle attribue une empreinte de 160 bits, à chaque message binaire de longueur inférieure à 2^{64} bits (ce qui est énorme, environ 1 milliard de gigaoctets). En voici une description succincte. Le document officiel la décrivant est le FIPS 180-1 [92] du NIST.

Le message m à traiter est d'abord complété pour obtenir un message M :

$$M = m \,\|\, 10\cdots0 \,\|\, \ell$$

où ℓ est la longueur de m exprimée en binaire sur 64 bits, et où le nombre de 0 est choisi dans l'intervalle $[0, 511]$ de telle façon que la longueur de M soit un multiple de 512 bits. Ensuite, M est divisé en blocs de 512 bits (soit 16 mots de 32 bits) :

$$M = M_1 \,\|\, \cdots \,\|\, M_n.$$

Voici l'algorithme correspondant à la fonction SHA-1 :

Entrée : Entier n et blocs M_1 à M_n de longueur 512.
Sortie : Empreinte de 160 bits.

$K_0 := 5A827999$, $K_1 := 6ED9EBA1$, $K_2 := 8F1BBCDC$, $K_3 := CA62C1D6$
$A := 67452301$, $B := EFCDAB89$, $C := 98BADCFE$, $D := 10325476$, $E := C3D2E1F0$
Pour i de 1 à n répéter
  Pour j de 0 à 15 définir W_j par $M_i = W_0 \,\|\, \cdots \,\|\, W_{15}$
  Pour j de 16 à 79 répéter
    $W_j := (W_{j-3} \oplus W_{j-8} \oplus W_{j-14} \oplus W_{j-16}) \lll 1$
  $A' := A$, $B' := B$, $C' := C$, $D' := D$, $E' := E$
  Pour j de 0 à 79 répéter
    $t := \lfloor i/20 \rfloor$
    $E' := A' \lll 5 + f_t(B', C', D') + E' + W_j + K_t$
    $(A', B', C', D', E') := (E', A', B' \lll 30, C', D')$
  $A := A + A'$, $B := B + B'$, $C := C + C'$, $D := D + D'$, $E := E + E'$
Retourner $A \,\|\, B \,\|\, C \,\|\, D \,\|\, E$

où les additions notées + s'entendent modulo 2^{32} et où le symbole $\lll$ désigne une rotation à gauche. Quand aux f_t, ce sont les fonctions booléennes suivantes, définies sur des mots de 32 bits :

$$f_0(B,C,D) = (B \wedge C) \vee (\overline{B} \wedge D)$$
$$f_1(B,C,D) = f_3(B,C,D) = B \oplus C \oplus D$$
$$f_2(B,C,D) = (B \wedge C) \vee (C \wedge D) \vee (D \wedge B)$$

Le NIST a normalisé de nouvelles fonctions de hachage, nommées SHA-256, SHA-384 et SHA-512, inspirées de SHA-1 mais produisant des empreintes plus longues (les noms correspondent à la longueur des empreintes en bits). Ces nouvelles fonctions sont décrites dans le draft FIPS 180-2.

Ripemd-160

Elle calcule elle aussi des empreintes de 160 bits et est dans sa construction assez similaire à SHA-1. Pour une description complète, voir `http://www.esat.kuleuven.ac.be/~bosselae/ripemd160.html`.

Fonction de hachage & logarithme discret

Bien que trop lente pour être performante en pratique, la fonction définie ci-après est une fonction de hachage dont la résistance forte aux collisions est prouvée. Précisément, elle repose sur le problème du logarithme discret dont l'énoncé est en 10.3.1.

2.1. — **Exemple.** Soit p un nombre premier tel que $q = (p-1)/2$ soit aussi premier (impair) et soient α, β deux générateurs (individuellement) de $(\mathbb{Z}/p\mathbb{Z})^*$. La fonction de Chaum/van Heijst/Pfitzmann est définie par

$$\begin{cases} \mathbb{Z}_q \times \mathbb{Z}_q \longrightarrow (\mathbb{Z}/p\mathbb{Z})^* \\ \quad (x,y) \longmapsto \alpha^x \beta^y \end{cases}$$

2.2. — **Théorème.** *Toute collision dans la fonction de Chaum/van Heijst/Pfitzmann permet de calculer* $\log_\alpha \beta$.

Démonstration — Soit $\alpha^x \beta^y = \alpha^{x'} \beta^{y'}$ avec $x, y, x', y' \in \mathbb{Z}_q$ et $(x,y) \neq (x',y')$ une telle collision. On a alors $\alpha^{x-x'} = \beta^{y'-y}$. Puisque $y, y' \in \mathbb{Z}_q$, on a $|y'-y| < q$ donc le pgcd de $y'-y$ avec $p-1$ divise 2. Si ce pgcd vaut 1, on peut calculer l'inverse z de $y'-y$ modulo $p-1$ et on a $\log_\alpha \beta = (x-x')z \bmod (p-1)$. Si le pgcd vaut 2, on se contente de calculer l'inverse z' de $y'-y$ modulo q. On a alors $\alpha^{(x-x')z'} = \pm\beta$ et donc le logarithme cherché est l'une des deux valeurs $(x-x')z' \bmod (p-1)$ et $(x-x')z' + q \bmod p-1$. Il est ensuite facile de trancher entre ces deux valeurs. □

Chapitre 5

Le groupe $(\mathbb{Z}/n\mathbb{Z})^*$

Une excellente référence pour ce chapitre est le livre de Ireland & Rosen [53].

Ordre du groupe $(\mathbb{Z}/n\mathbb{Z})^*$

On note φ la fonction indicatrice d'Euler, i.e $\varphi(1) = 1$ et, pour $n \geqslant 2$ entier, l'entier $\varphi(n)$ est le nombre d'éléments du groupe $(\mathbb{Z}/n\mathbb{Z})^*$. On voit facilement que la fonction φ vérifie les propriétés suivantes :

$$\begin{aligned} \varphi(p^r) &= p^{r-1}(p-1) && \text{pour } p \text{ premier et } r \in \mathbb{N}^* \\ \varphi(n_1 n_2) &= \varphi(n_1)\varphi(n_2) && \text{pour } n_1, n_2 \in \mathbb{N}^* \text{ premiers entre eux.} \end{aligned}$$

0.1. — Proposition. *Pour $n \in \mathbb{N}^*$, on a $\sum_{d|n} \varphi(d) = n$.*

Démonstration — Ecrire les fractions i/n sous forme réduite, pour $0 \leqslant i < n$. Pour d diviseur de n, les fractions obtenues de dénominateur d sont les j/d avec $0 \leqslant j < d$ et $\mathrm{pgcd}(j, d) = 1$. Il y en a donc $\varphi(d)$. La formule est alors claire. □

1. Structure du groupe $(\mathbb{Z}/n\mathbb{Z})^*$

Si p est un nombre premier, l'anneau $\mathbb{Z}/p\mathbb{Z}$ est un corps (le théorème de Bézout montre que tout élément non nul est inversible). L'ordre de son groupe des unités est donc $p-1$. Nous commençons par rappeler le résultat très important selon lequel ce groupe est cyclique (ceci se généralise à tout groupe multiplicatif fini dans un corps).

Le cas où n est premier

1.1. — Lemme. *Soient p un nombre premier et $d \in \mathbb{N}^*$. Le nombre d'éléments d'ordre d dans le groupe $(\mathbb{Z}/p\mathbb{Z})^*$ est 0 ou $\varphi(d)$.*

Démonstration — Supposons qu'il y ait au moins un élément a d'ordre d. Les a^i pour $0 \leqslant i < d$ sont solutions de l'équation $x^d - 1 = 0$, qui ne peut avoir d'autres solutions puisqu'elle est de degré d. Les éléments d'ordre d sont donc à chercher parmi les a^i et on voit que ce sont ceux pour lesquels $\mathrm{pgcd}(i, d) = 1$ (car l'ordre de a^i est $d/pgcd(d, i)$). Ils sont bien au nombre de $\varphi(d)$. □

1.2. — Théorème. *Si p est un nombre premier, le groupe $(\mathbb{Z}/p\mathbb{Z})^*$ est cyclique.*

Démonstration — Pour $d \in \mathbb{N}^*$, notons $\psi(d)$ le nombre d'éléments d'ordre d. On a alors, puisque $\psi(d)$ est nul lorsque $d \nmid p-1$,

$$\sum_{d|p-1} \psi(d) = \sum_{d \in \mathbb{N}^*} \psi(d) = p - 1 = \sum_{d|p-1} \varphi(d). \tag{1}$$

D'autre part, le lemme 1.1 fournit l'inégalité $\psi(d) \leqslant \varphi(d)$. Afin de satisfaire l'équation (1), il est nécessaire qu'on ait l'égalité $\psi(d) = \varphi(d)$ pour $d \mid p-1$. En particulier, on a $\psi(p-1) = \varphi(p-1) \geqslant 1$ ce qui assure l'existence d'un générateur. □

1.3. — Exercice. Combien y a-t-il de générateurs distincts du groupe $(\mathbb{Z}/13\mathbb{Z})^*$? Les énumérer.

1.4. — Exercice. Soit p un nombre premier et a un élément d'ordre 3 dans $(\mathbb{Z}/p\mathbb{Z})^*$. Déterminer l'ordre de $1 + a$.

Cas d'une puissance d'un premier

1.5. — Lemme. *Soient p un nombre premier et $a, b \in \mathbb{Z}$ tels que $a \equiv b$ modulo p. Alors, pour $s \in \mathbb{N}$,*

$$a^{p^s} \equiv b^{p^s} \pmod{p^{s+1}}.$$

DÉMONSTRATION — C'est trivialement vrai pour $s = 0$. Posons $u = a^{p^{s-1}}$, $v = b^{p^{s-1}}$ et supposons la propriété vraie au rang $s-1$, c'est-à-dire $u \equiv v$ modulo p^s. On a l'identité

$$a^{p^s} - b^{p^s} = u^p - v^p = (u-v)(u^{p-1} + u^{p-2}v + \cdots + v^{p-1}).$$

Mais on a $u \equiv a \equiv b \equiv v$ modulo p, donc p divise le facteur $(u^{p-1}+u^{p-2}v+\cdots+v^{p-1})$. Comme p^s divise $u-v$, on voit que p^{s+1} divise $u^p - v^p$. Le résultat s'en suit par récurrence. □

Le cas d'une puissance d'un premier impair

1.6. — Lemme. *Soient p un nombre premier impair et $a \in \mathbb{Z}$. Alors, pour $s \in \mathbb{N}^*$,*

$$(1+ap)^{p^{s-1}} \equiv 1 + ap^s \pmod{p^{s+1}}.$$

DÉMONSTRATION — C'est trivial pour $s = 1$. Supposons que c'est vrai au rang s. Alors, en appliquant le lemme 1.5, on obtient

$$\begin{aligned}(1+ap)^{p^s} \equiv (1+ap^s)^p &\equiv \sum_{k=0}^{p} C_p^k a^k p^{ks} \pmod{p^{s+2}} \\ &\equiv 1 + pap^s + \frac{p(p-1)}{2}a^2p^{2s} \qquad \text{puisque } ks \geqslant s+2 \text{ pour } k \geqslant 3 \\ &\equiv 1 + ap^{s+1} \qquad \text{puisque } 2s \geqslant s+1.\end{aligned}$$

On a donc le résultat par récurrence. □

1.7. — Théorème. *Si p est un nombre premier impair et $r \in \mathbb{N}^*$, le groupe $(\mathbb{Z}/p^r\mathbb{Z})^*$ est cyclique.*

DÉMONSTRATION — D'après le théorème 1.2, on peut supposer que $r \geqslant 2$. Soit g un entier d'ordre $p-1$ modulo p. On a

$$(g+p)^{p-1} \equiv g^{p-1} + (p-1)g^{p-2}p \equiv g^{p-1} - pg^{p-2} \not\equiv g^{p-1} \pmod{p^2}.$$

Quitte à remplacer g par $g+p$, on peut donc supposer que $g^{p-1} \not\equiv 1$ modulo p^2 et poser $g^{p-1} = 1+ap$ avec $p \nmid a$. Soit e l'ordre de g modulo p^r. C'est un diviseur de $\varphi(p^r) = p^{r-1}(p-1)$ et un multiple de l'ordre $p-1$ de g modulo p. Donc, e est de la forme $p^s(p-1)$ avec $0 \leqslant s \leqslant r-1$. Mais, d'après le lemme 1.6,

$$g^{p^{r-2}(p-1)} = (1+ap)^{p^{r-2}} \equiv 1 + ap^{r-1} \not\equiv 1 \pmod{p^r}.$$

Donc $e = p^{r-1}(p-1)$ et g engendre $(\mathbb{Z}/p^r\mathbb{Z})^*$. □

Le cas d'une puissance de 2

1.8. — Lemme. *Pour chaque entier $s \geqslant 3$, on a*

$$5^{2^{s-3}} \equiv 1 + 2^{s-1} \pmod{2^s}.$$

DÉMONSTRATION — C'est trivial pour $s = 3$. Supposons la relation vraie au rang s. Alors, par le lemme 1.5

$$5^{2^{s-2}} \equiv (1+2^{s-1})^2 \equiv 1 + 2^s + 2^{2s-2} \equiv 1 + 2^s \pmod{2^{s+1}}.$$

On a donc le résultat par récurrence. □

1.9. — Théorème. *Pour $r \in \mathbb{N}^*$, le groupe $(\mathbb{Z}/2^r\mathbb{Z})^*$ est isomorphe à*

$$\begin{aligned} \{0\} \quad & \text{si } r = 1, \\ \mathbb{Z}/2\mathbb{Z} \quad & \text{si } r = 2, \\ \mathbb{Z}/2\mathbb{Z} \times \mathbb{Z}/2^{r-2}\mathbb{Z} \quad & \text{si } r \geqslant 3. \end{aligned}$$

DÉMONSTRATION — Seul le cas $r \geqslant 3$ mérite une explication. D'après le lemme 1.8, on a

$$5^{2^{r-3}} \not\equiv 1 \pmod{2^r} \qquad \text{mais} \qquad 5^{2^{r-2}} \equiv 1 \pmod{2^r}.$$

L'ordre de 5 modulo 2^r est donc 2^{r-2}.

Montrons que -1 et 5 engendrent le groupe $(\mathbb{Z}/2^r\mathbb{Z})^*$. Si $a, a', b, b' \in \mathbb{Z}$ vérifient $(-1)^a 5^b \equiv (-1)^{a'} 5^{b'}$ modulo 2^r alors on a $(-1)^a \equiv (-1)^{a'}$ modulo 4. Donc a et a' ont même parité. On a alors $5^b \equiv 5^{b'}$ modulo 2^r et, puisque l'ordre de 5 est 2^{r-2}, on obtient $b \equiv b'$ modulo 2^{r-2}. Le sous-groupe de $(\mathbb{Z}/2^r\mathbb{Z})^*$ engendré par -1 et 5 est donc de cardinal $2 \times 2^{r-2} = 2^{r-1} = \varphi(2^r)$. Ce sous-groupe est donc $(\mathbb{Z}/2^r\mathbb{Z})^*$ tout entier.

Enfin, pour $a, b \in \mathbb{Z}$, l'entier $(-1)^a 5^b$ élevé à la puissance 2^{r-2} est congru à 1 modulo 2^r. L'ordre de chaque élément du groupe $(\mathbb{Z}/2^r\mathbb{Z})$ est donc diviseur de 2^{r-2} et ce groupe n'est pas cyclique. □

1.10. — Exercice. Montrer que le produit de deux groupes cycliques d'ordres respectifs a et b est cyclique si et seulement si a et b sont premiers entre eux.

Le cas général

Ces différents résultats combinés avec le théorème chinois donnent les deux théorèmes suivants.

1.11. — Théorème. *Soit un entier $n \geqslant 2$ et $n = 2^r p_1^{r_1} \cdots p_k^{r_k}$ sa décomposition en facteurs premiers (avec les p_i distincts et impairs et $r_i > 0$). Pour $r = 0$ ou 1, le groupe $(\mathbb{Z}/n\mathbb{Z})^*$ se décompose en produit de k groupes cycliques d'ordres respectifs $p_k^{r_k-1}(p_k - 1)$. Pour $r = 2$, s'ajoute à ces k composantes un facteur isomorphe à $(\mathbb{Z}/2\mathbb{Z})$. Pour $r \geqslant 3$, s'ajoute aux $k+1$ composantes déjà citées un facteur isomorphe à $(\mathbb{Z}/2^{r-2}\mathbb{Z})$.* □

1.12. — Corollaire. *Soit un entier $n \geqslant 2$. Le groupe $(\mathbb{Z}/n\mathbb{Z})^*$ est cyclique si et seulement si n vaut 2 ou 4 ou est de la forme p^r ou $2p^r$ avec p premier impair et $r \geqslant 1$.*

1.13. — Définition. Soit $n \geqslant 2$ un entier. On dit que $a \in \mathbb{Z}$ est un *élément primitif modulo n* si sa classe engendre le groupe $(\mathbb{Z}/n\mathbb{Z})^*$.

1.14. — Exercice. Déterminer des éléments primitifs modulo 11, modulo 22, modulo 121, et modulo 1331.

1.15. — Exercice. Soit $n \geqslant 2$ entier. Quel est le nombre de solutions modulo n de l'équation $x^{\varphi(n)} \equiv 1$?

1.16. — Exercice. Soient $a, n \in \mathbb{N}^*$ tels que $m = a^n - 1 \geqslant 2$. Montrer que $n \mid \varphi(m)$.

Indicateur de Carmichael

1.17. — Définitions. Soit G un groupe (multiplicatif) et $e \in \mathbb{N}^*$. On dit que e est **un** *exposant* de G si on a $g^e = 1$ pour tout $g \in G$. On dit que e est **l'***exposant* de G, noté $\lambda(G)$, si e est le plus petit entier vérifiant cette propriété.

1.18. — Proposition. *Soit G un groupe abélien fini. Alors l'exposant de G existe et est un diviseur de l'ordre de G. De plus, G contient un élément d'ordre $\lambda(G)$.*

DÉMONSTRATION — Exercice. On peut utiliser la décomposition de G en produit de groupes cycliques. □

1.19. — Définition. Soit $m \geqslant 2$ entier. On désigne par $\lambda(m)$ l'exposant du groupe $(\mathbb{Z}/m\mathbb{Z})^*$. La fonction λ ainsi définie (on pose en plus $\lambda(1) = 1$) s'appelle l'*indicateur de Carmichael*.

En vertu du théorème 1.11, l'indicateur de Carmichael satisfait les relations suivantes.

$$\begin{aligned} &\lambda(2) = 1, \quad \lambda(4) = 2, \quad \lambda(2^r) = 2^{r-2} \quad \text{pour } r \geqslant 3 \\ &\lambda(p^r) = p^{r-1}(p-1) \qquad \text{pour } p \text{ premier impair et } r \in \mathbb{N}^*, \\ &\lambda(n_1 n_2) = \operatorname{ppcm}\big(\lambda(n_1)\lambda(n_2)\big), \qquad \text{pour } n_1, n_2 \in \mathbb{N}^* \text{ premiers entre eux.} \end{aligned}$$

Généralisations du petit théorème de Fermat

On rappelle le petit théorème de Fermat et l'identité d'Euler qui résultent de la structure de groupe de $(\mathbb{Z}/n\mathbb{Z})^*$.

1.20. — Théorème (Petit théorème de Fermat). *Soient p un nombre premier et a un entier tel que $p \nmid a$. On a $a^{p-1} \equiv 1$ modulo p.*

1.21. — Théorème (Identité d'Euler). *Soient $n \geqslant 2$ un entier et a tel que $\mathrm{pgcd}(a,n) = 1$. Alors $a^{\varphi(n)} \equiv 1$ modulo n.*

Nous utiliserons dans le chapitre *Primalité* la version forte suivante du petit théorème de Fermat.

1.22. — Théorème. *Soient p un premier impair et a un entier tel que $p \nmid a$. On a l'une des assertions suivantes, en posant $p-1 = 2^k q$ avec q impair,*

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

ou bien

$$\text{il existe un entier } i \text{ tel que } 0 \leqslant i < k \text{ et } a^{2^i q} \equiv -1 \pmod{p}.$$

Démonstration — Utiliser 1.20 et le fait que 1 n'ait que deux racines carrées ± 1 dans le corps $\mathbb{Z}/p\mathbb{Z}$. □

2. Carrés modulo un nombre premier

2.1. — Proposition. *Soit p un nombre premier. L'ensemble des carrés de $(\mathbb{Z}/p\mathbb{Z})^*$ (l'image de l'application $x \mapsto x^2$) forme un sous-groupe d'indice 2 dans $(\mathbb{Z}/p\mathbb{Z})^*$.*

Démonstration — Car le noyau de l'application $x \mapsto x^2$ est $\{\pm 1\}$. □

Symbole de Legendre

2.2. — Définition. Soient p un nombre premier impair et a un entier non divisible par p. On définit le *symbole de Legendre* noté (a/p) par

$$\left(\frac{a}{p}\right) = \begin{cases} 1 & \text{si } a \text{ est un carré modulo } p, \\ -1 & \text{si } a \text{ n'est pas un carré modulo } p. \end{cases}$$

2.3. — Proposition. *Soient p un nombre premier impair et a un entier non divisible par p. On a*

$$\left(\frac{a}{p}\right) \equiv a^{\frac{p-1}{2}} \pmod{p}.$$

Démonstration — L'anneau $(\mathbb{Z}/p\mathbb{Z})$ est un corps. Le groupe $(\mathbb{Z}/p\mathbb{Z})^*$ est donc cyclique d'après 1.2 et 1 n'y a que deux racines carrées ± 1. Soit g un élément primitif modulo p. On a $g^{(p-1)/2} \equiv -1$ modulo p puisque g est d'ordre $p-1$ modulo p. L'entier a est congru modulo p à un g^k et on a

$$a^{\frac{p-1}{2}} \equiv (g^k)^{\frac{p-1}{2}} \equiv (-1)^k \pmod{p}.$$

Mais a est un carré modulo p si et seulement si k est pair, d'après 2.1. □

2.4. — Proposition. *Soient p un nombre premier impair et a, b deux entiers non divisibles par p. On a les identités*

$$\left(\frac{a}{p}\right) = \left(\frac{b}{p}\right) \qquad \text{si } a \equiv b \pmod{p}$$

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

Démonstration — Immédiat d'après 2.3. □

Loi de réciprocité quadratique

2.5. — Lemme (Gauss). *Soient p un nombre premier impair et a un entier non divisible par p. Soit s le nombre d'entiers $ja \bmod p$ avec $1 \leqslant j \leqslant (p-1)/2$ supérieurs à $p/2$. Alors $(a/p) = (-1)^s$.*

DÉMONSTRATION — Notons $u_1, \ldots, u_s$ les entiers $ja \bmod p$ supérieurs à $p/2$ et $v_1, \ldots, v_t$ ceux qui sont inférieurs à $p/2$ ($1 \leqslant j \leqslant (p-1)/2$). On a donc $s + t = (p-1)/2$. On ne peut avoir $p - u_i = v_{i'}$ pour aucun couple (i, i') car on aurait alors une congruence $-ja \equiv j'a$ modulo p, donc $-j \equiv j'$ modulo p, ce qui est impossible pour $1 \leqslant j, j' \leqslant (p-1)/2$. On a donc

$$\{p - u_i \mid 1 \leqslant i \leqslant s\} \cup \{v_i \mid 1 \leqslant i \leqslant t\} = \{1, \ldots, \frac{p-1}{2}\}. \tag{2}$$

Ainsi,

$$\begin{aligned}\left(\frac{p-1}{2}\right)! &\equiv \prod_{i=1}^{s}(p-u_i)\prod_{i=1}^{t} v_i \equiv (-1)^s \prod_{i=1}^{s} u_i \prod_{i=1}^{t} v_i \\ &\equiv (-1)^s \prod_{j=1}^{(p-1)/2} (ja) \equiv (-1)^s \left(\frac{p-1}{2}\right)! \left(\frac{a}{p}\right) \pmod p\end{aligned}$$

d'après 2.3. D'où le résultat. □

2.6. — Lemme. *Soient p un nombre premier impair et a un entier non divisible par p. On désigne par s l'entier défini dans le lemme précédent et on pose $M = M(a,p) = \sum_{j=1}^{(p-1)/2} \left\lfloor \frac{ja}{p} \right\rfloor$. Alors $s \equiv (p^2-1)/8$ modulo 2 si $a = 2$, et $s \equiv M$ modulo 2 si a est impair.*

DÉMONSTRATION — On a d'une part

$$\sum_{j=1}^{(p-1)/2} ja = p \sum_{j=1}^{(p-1)/2} \left\lfloor \frac{ja}{p} \right\rfloor + \sum_{i=1}^{s} u_i + \sum_{i=1}^{t} v_i = pM + \sum_{i=1}^{s} u_i + \sum_{i=1}^{t} v_i.$$

D'autre part, d'après (2),

$$\sum_{j=1}^{(p-1)/2} j = \sum_{i=1}^{s}(p-u_i) + \sum_{i=1}^{t} v_i = sp - \sum_{i=1}^{s} u_i + \sum_{i=1}^{t} v_i.$$

Par différence, on obtient

$$(a-1) \sum_{j=1}^{(p-1)/2} j = p(M-s) + 2\sum_{i=1}^{s} u_i.$$

Si a est impair, on voit alors que $M - s$ doit être pair. Si $a = 2$ alors les ja sont inférieurs à p donc $M = 0$. Il vient

$$-ps \equiv \sum_{j=1}^{(p-1)/2} j \equiv \frac{p^2-1}{8} \pmod 2$$

et résultat est montré. □

2.7. — Théorème (Réciprocité quadratique). *Soient p, q deux nombres premiers impairs, on a la relation*

$$\left(\frac{p}{q}\right)\left(\frac{q}{p}\right) = (-1)^{\frac{p-1}{2}\frac{q-1}{2}}.$$

DÉMONSTRATION — Dans le plan cartésien, $M(q,p)$ est le nombre de points à coordonnées entières dans le triangle délimité par l'axe des abscisses, la droite verticale d'ordonnée $p/2$ et la droite passant par l'origine et de pente q/p. Par symétrie, $M(p,q)$ est le nombre de points à coordonnées entières dans le triangle

délimité par l'axe des ordonnées, la droite horizontale d'ordonnée $q/2$ et la même droite de pente q/p que précédemment. Au total $M(p,q)+M(q,p)$ est le nombre de points à coordonnées entières dans le rectangle $]0,p/2]\times]0,q/2]$. Ils sont au nombre de $(p-1)/2\cdot(q-1)/2$. Mais d'après les deux lemmes,

$$\left(\frac{p}{q}\right)\left(\frac{q}{p}\right)=(-1)^{M(p,q)+M(q,p)}$$

d'où le résultat. □

2.8. — Théorème (lois complémentaires). *Soit p un nombre premier impair. On a les relations*

$$\left(\frac{-1}{p}\right)=(-1)^{(p-1)/2},\qquad \left(\frac{2}{p}\right)=(-1)^{(p^2-1)/8}.$$

Démonstration — La première loi est une conséquence immédiate de 2.3. D'autre part, d'après le lemme de Gauss, on a $(2/p)=(-1)^s$ (avec s défini dans 2.5). La deuxième loi résulte donc du cas $a=2$ du lemme 2.6. □

3. Carrés modulo un entier quelconque

Symbole de Jacobi

On étend la définition du symbole de Legendre aux "dénominateurs" non nécessairement premiers, mais impairs.

3.1. — Définition. Soient m,a deux entiers, avec $m>0$ impair et soit $m=\prod_{i=1}^k p_i^{\alpha_i}$ la décomposition de m en facteurs premiers. On définit le *symbole de Jacobi*, noté (a/m) par

$$\left(\frac{a}{m}\right)=\begin{cases}0 & \text{si } pgcd(a,m)>1,\\ \displaystyle\prod_{i=1}^k\left(\frac{a}{p_i}\right)^{\alpha_i} & \text{si } pgcd(a,m)=1.\end{cases}$$

3.2. — Proposition. *Soient m,n,a,b quatre entiers, avec $m,n>0$ impairs. On a les identités*

$$\left(\frac{a^2}{m}\right)=1 \qquad \textit{si } \mathrm{pgcd}(a,m)=1$$
$$\left(\frac{a}{m}\right)=\left(\frac{b}{m}\right) \qquad \textit{si } a\equiv b \pmod m$$
$$\left(\frac{a}{m}\right)\left(\frac{b}{m}\right)=\left(\frac{ab}{m}\right)$$
$$\left(\frac{a}{m}\right)\left(\frac{a}{n}\right)=\left(\frac{a}{mn}\right)$$

Démonstration — Exercice. □

Loi de réciprocité quadratique (bis)

3.3. — Lemme. *Soient p,q des entiers impairs. Alors, les entiers $(p-1)/2+(q-1)/2$ et $(pq-1)/2$ ont même parité.*

Démonstration — On a $4|(p-1)(q-1)=(pq-1)-(p-1)-(q-1)$, donc $(p-1)+(q-1)\equiv pq-1$ modulo 4. □

La loi de réciprocité quadratique s'étend sans modification aux entiers non nécessairement premiers mais impairs.

3.4. — Théorème (Réciprocité quadratique). *Soient $m, n \in \mathbb{N}^*$ impairs et premiers entre eux, on a la relation*

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

DÉMONSTRATION — Peut se déduire par multiplicativité du cas premier 2.7, en utilisant 3.3. Certaines démonstrations de la réciprocité quadratique donnent directement le résultat sous cette forme générale. □

3.5. — Théorème (lois complémentaires). *Soit m un nombre impair. On a les relations*

$$\left(\frac{-1}{m}\right) = (-1)^{(m-1)/2}, \qquad \left(\frac{2}{m}\right) = (-1)^{(m^2-1)/8}.$$

DÉMONSTRATION — Exercice. Le déduire par multiplicativité du cas premier 2.8. □

3.6. — Exercice. Calculer le symbole de Jacobi (383/443).

3.7. — Exercice. Montrer la forme plus générale suivante de la loi de réciprocité quadratique. Soient $m, n \in \mathbb{Z}$ impairs et premiers entre eux, on a

$$\left(\frac{m}{|n|}\right)\left(\frac{n}{|m|}\right) = \varepsilon(-1)^{\frac{m-1}{2}\frac{n-1}{2}} \qquad \text{avec } \varepsilon = -1 \text{ si } m \text{ et } n \text{ sont négatifs, } \varepsilon = 1 \text{ sinon.}$$

3.8. — Exercice. Montrer que, pour p premier impair, 6 est un carré modulo p si et seulement si $p \equiv \pm 1$ ou $p \equiv \pm 5$ modulo 24.

Nombre de racines carrées modulo n

3.9. — Lemme. *Soient p un nombre premier impair, $r \in \mathbb{N}^*$ et $a \in \mathbb{Z}$ tel que $p \nmid a$. Si x vérifie $x^2 \equiv a$ modulo p^r alors, il existe x', unique modulo p^{2r}, tel que $x' \equiv x$ modulo p^r et $x'^2 \equiv a$ modulo p^{2r}.*

DÉMONSTRATION — Un tel x' doit être de la forme $x' = x + p^r y$. Mais alors $x'^2 \equiv x^2 + 2p^r xy$ modulo p^{2r} et on a $x'^2 \equiv a$ modulo p^{2r} si et seulement si $\frac{x^2-a}{p^r} + 2xy \equiv 0$ modulo p^r, c'est-à-dire $y \equiv \frac{a-x^2}{p^r}(2x)^{-1}$ modulo p^r. L'existence et l'unicité de y modulo p^r implique celle de x' modulo p^{2r}. □

3.10. — Théorème. *Soient p un nombre premier impair, $r \in \mathbb{N}^*$ et $a \in \mathbb{Z}$ tel que $p \nmid a$. Le nombre de solutions modulo p^r de l'équation $x^2 \equiv a$ est $1 + (a/p)$.*

DÉMONSTRATION — Clair par le lemme précédent. □

3.11. — Théorème. *Soient $r \in \mathbb{N}^*$ et $a \in \mathbb{Z}$ impair. Le nombre de solutions modulo 2^r de l'équation $x^2 \equiv a$ est donné par :*

- *Si $r = 1$, toujours une solution.*
- *Si $r = 2$, deux solutions lorsque $a \equiv 1$ modulo 4, zéro lorsque $a \equiv 3$ modulo 4.*
- *Si $r \geqslant 3$, quatre solutions lorsque $a \equiv 1$ modulo 8, zéro sinon.*

DÉMONSTRATION — Seul le troisième point mérite une explication. Si l'équation a une solution alors c'est un entier impair $2k+1$. Mais

$$(2k+1)^2 = 4k(k+1) + 1 \equiv 1 \pmod 8$$

puisque $k(k+1)$ est pair. Donc $a \equiv 1$ modulo 8.

Inversement, supposons que $a \equiv 1$ modulo 8. Montrons par récurrence sur r que l'équation $x^2 = a$ modulo 2^r a des solutions. C'est vrai pour $r = 3$ (les classes de 1, 3, 5 et 7 sont toutes quatre solutions). Supposons qu'il y ait une solution au rang $r-1$ où $r \geqslant 4$, soit $x^2 \equiv a$ modulo 2^{r-1}. Alors, pour $y \in \mathbb{Z}$, on a

$$(x + 2^{r-2}y)^2 \equiv x^2 + 2^{r-1}xy \equiv x^2 + 2^{r-1}y \pmod{2^r}.$$

En prenant $y = (a - x^2)/2^{r-1} \bmod 2$, on obtient une solution $x + 2^{r-2}y$ modulo 2^r. On a donc ainsi montré l'existence de solutions quel que soit $r \geqslant 3$. On a même quatre solutions puisque si x en est une, alors $\pm x$ et $\pm x + 2^{r-1}$ en sont quatre.

Montrons qu'il y a seulement quatre solutions. Soient x et y deux solutions. On a alors $x^2 \equiv y^2$ modulo 2^r et donc $2^r \mid (x+y)(x-y)$. Puisque x et y sont impairs, on peut écrire $2^{r-2} \mid \frac{x+y}{2}\frac{x-y}{2}$. Mais $\frac{x+y}{2}$ et $\frac{x-y}{2}$ ne peuvent être tous les deux pairs puisque leur somme vaut x. Donc 2^{r-2} divise l'un des deux. On obtient alors $y \equiv \pm x$ ou $y \equiv \pm x + 2^{r-1}$ modulo 2^r et il n'y a que quatre solutions. □

Des théorèmes ci-dessus et du théorème chinois, on peut déduire les deux résultats suivants.

3.12. — Théorème. *Soient $n \geqslant 2$ et $a \in \mathbb{Z}$ premiers entre eux. Alors a est un carré modulo n si et seulement si*

$$\begin{cases} \left(\dfrac{a}{p}\right) = 1 & \text{pour tout diviseur premier impair } p \text{ de } n, \text{ et} \\ a \equiv 1 \pmod 4 & \text{si } n \equiv 4 \pmod 8 \\ a \equiv 1 \pmod 8 & \text{si } 8 \mid n. \end{cases}$$

3.13. — Théorème. *Soient $n \geqslant 2$ et $a \in \mathbb{Z}$ premiers entre eux. Le nombre de racines carrées modulo n est donné par*

$$w \prod_{\substack{p \mid n \\ p \text{ premier impair}}} \left(1 + \left(\frac{a}{p}\right)\right) \qquad \text{où } w = \begin{cases} 1 & \text{si } 4 \nmid n \\ 2 & \text{si } n \equiv 4 \text{ modulo 8 et } a \equiv 1 \text{ modulo 4} \\ 4 & \text{si } 8 \mid n \text{ et } a \equiv 1 \text{ modulo 8} \\ 0 & \text{sinon.} \end{cases}$$

3.14. — Exercice. Résoudre l'équation $x^2 \equiv 2$ modulo 5831.

Entiers de Blum

3.15. — Définition. Un *entier de Blum* est un produit de deux nombres premiers distincts congrus à 3 modulo 4.

3.16. — Proposition. *Soient n un entier de Blum et a un carré modulo n (avec $\operatorname{pgcd}(a, n) = 1$). Alors, a admet quatre racines carrées modulo n dont exactement une est elle-même un carré modulo n.*

Démonstration — Exercice. □

3.17. — Définition. Soient n un entier de Blum et a un carré modulo n (avec $\operatorname{pgcd}(a, n) = 1$). L'unique racine carrée de a modulo n qui soit elle-même un carré modulo n est appelée la *racine principale de a modulo n*.

3.18. — Proposition. *Soit $n = pq$ un entier de Blum. L'application $x \mapsto x^2$ est une permutation sur l'ensemble des carrés modulo n (relativement premiers à n). Sa réciproque peut être exprimée par*

$$y \longmapsto y^{\left((p-1)(q-1)+4\right)/8}.$$

Démonstration — Exercice. □

Résidualité quadratique

3.19. — Problème. Etant donné un entier impair n, posons

$$\mathbb{Z}_n^+ = \left\{ x \in \mathbb{Z}_n \,\middle|\, \left(\frac{x}{n}\right) = 1 \right\}, \qquad Q = \{x \in \mathbb{Z}_n \mid x = y^2 \bmod n \text{ avec } y \in \mathbb{Z}_n^*\}, \qquad \overline{Q} = \mathbb{Z}_n^+ \setminus Q. \tag{3}$$

Il est clair que $Q \subseteq \mathbb{Z}_n^+$. Mais en général (lorsque n n'est pas une simple puissance d'un nombre premier), cette inclusion est stricte. Le *problème de la résidualité quadratique* est celui de savoir distinguer, parmi les éléments de $\mathbb{Z}_n^+$, ceux qui sont dans Q de ceux de $\overline{Q}$.

Les éléments x de Q sont ceux vérifiant $(x/p) = 1$ pour tout diviseur premier p de n. Si on sait factoriser n alors on sait résoudre le problème de la résidualité quadratique pour n. Inversement, on pense généralement que le problème de la résidualité quadratique est aussi difficile que celui de la factorisation, mais rien n'a été démontré dans ce sens. Ce problème a plusieurs applications cryptographiques, en particulier lorsque n est un entier de type RSA (un produit de deux grands nombres premiers distincts).

Chapitre 6

Un peu plus d'arithmétique

1. Corps finis

Cette section sera très peu développée ici car les corps finis seront étudiés plus en détail dans le cours de codage. Nous rappelons tout de même quelques résultats essentiels. Le livre de Lidl & Niederreiter [71] est une référence complète.

1.1. — **Théorème.** *Pour chaque puissance p^r d'un nombre premier, il existe un corps fini, unique à isomorphisme près, de cardinal p^r. Nous le noterons $\mathbb{F}_{p^r}$.*

Le corps fini $\mathbb{F}_{p^r}$ est de caractéristique p. On peut le construire en choisissant un polynôme f de degré r irréductible sur $\mathbb{Z}/p\mathbb{Z}$ (il en existe toujours). L'anneau quotient $\mathbb{Z}/p\mathbb{Z}[X]/(f)$ est alors un corps fini à p^r éléments. C'est un espace vectoriel sur le corps $\mathbb{F}_p = \mathbb{Z}/p\mathbb{Z}$ et la famille $\{1, X, \ldots, X^{n-1}\}$ en est une base.

1.2. — **Théorème.** *Pour chaque corps fini $\mathbb{F}_{p^r}$, le groupe multiplicatif $\mathbb{F}_{p^r}^*$ est cyclique.*

2. Corps quadratiques

Pour une introduction aux corps quadratiques et aux discriminants, on pourra consulter le petit livre de Samuel [116].

Un *corps quadratique* est une extension de degré 2 du corps $\mathbb{Q}$ des nombres rationnels. C'est donc un corps de la forme $\mathbb{Q}(\alpha)$ où α est racine d'un polynôme $aX^2 + bX + c$ irréductible de degré 2, avec $a, b, c \in \mathbb{Z}$, $a \neq 0$.

On peut toujours poser $\delta = 2a\alpha + b$ et $d = b^2 - 4ac$. On a alors $\mathbb{Q}(\delta) = \mathbb{Q}(\alpha)$ et le polynôme minimal de δ sur $\mathbb{Q}$ est $X^2 - d$. Les corps quadratiques sont donc les corps de la forme $\mathbb{Q}(\sqrt{d})$, où $d \in \mathbb{Z}$ n'est pas un carré parfait.

Si $d > 0$ alors $\mathbb{Q}(\sqrt{d})$ est contenu dans $\mathbb{R}$ et on dit que c'est un corps quadratique *réel*. Si $d < 0$ alors $\mathbb{Q}(\sqrt{d})$ n'est pas contenu dans $\mathbb{R}$ et on dit que c'est un corps quadratique *imaginaire*.

Soit $K = \mathbb{Q}(\sqrt{d})$ un corps quadratique. Il possède un unique automorphisme non trivial σ et, pour $\alpha = a + b\sqrt{d} \in K$ avec $a, b \in \mathbb{Q}$, le conjugué de α, noté $\overline{\alpha}$, est $\sigma(\alpha) = a - b\sqrt{d}$. La norme de α est $N(\alpha) = \alpha\overline{\alpha} = a^2 - db^2$ et sa trace est $\alpha + \overline{\alpha} = 2a$.

Entiers quadratiques

Un *entier algébrique* (sur $\mathbb{Q}$) est une racine d'un polynôme unitaire, à coefficients dans $\mathbb{Z}$, et irréductible sur $\mathbb{Q}$. Lorsque ce polynôme est de degré 2, on parle d'*entier quadratique*. L'ensemble $\mathbb{Z}_K$ des entiers algébriques contenus dans un corps quadratique $K = \mathbb{Q}(\sqrt{d})$ est un anneau (l'*anneau des entiers de K*) contenant $\mathbb{Z}[\sqrt{d}]$. Toutefois, l'anneau $\mathbb{Z}[\sqrt{d}]$ est parfois strictement contenu dans $\mathbb{Z}_K$ (dans ce cas, le groupe $\mathbb{Z}[\sqrt{d}]$ est d'indice 2 dans $\mathbb{Z}_K$). Précisément on a le résultat suivant.

2.1. — **Théorème.** *Soit $K = \mathbb{Q}(\sqrt{d})$ un corps quadratique, avec d sans facteur carré. Son anneau des entiers est donné par*

$$\mathbb{Z}_K = \begin{cases} \mathbb{Z}[\sqrt{d}] = \{a + b\sqrt{d} \mid a, b \in \mathbb{Z}\} & \text{si } d \equiv 2, 3 \text{ modulo } 4 \\ \mathbb{Z}\left[\dfrac{-1+\sqrt{d}}{2}\right] = \{\dfrac{1}{2}(a + b\sqrt{d}) \mid a, b \in \mathbb{Z}, a \equiv b \pmod 2\} & \text{si } d \equiv 1 \text{ modulo } 4. \end{cases}$$

Démonstration — Soit $\alpha = a + b\sqrt{d}$ un élément de K (avec $a, b \in \mathbb{Q}$). On a clairement $\mathbb{Z}_K \cap \mathbb{Q} = \mathbb{Z}$, donc on peut supposer que $\alpha \notin \mathbb{Z}$. Son polynôme minimal sur $\mathbb{Q}$ est alors $(X - \alpha)(X - \overline{\alpha}) = X^2 - 2aX + a^2 - db^2$

donc α est entier si et seulement si $2a$ et $a^2 - db^2$ sont dans $\mathbb{Z}$. Pour cela, deux cas sont possibles. Dans le premier, $a \in \mathbb{Z}$, et on doit avoir $db^2 \in \mathbb{Z}$, ce qui est équivalent à $b \in \mathbb{Z}$ puisque d est sans facteur carré. Dans le deuxième cas, $a \in \frac{1}{2}\mathbb{Z} \setminus \mathbb{Z}$. On a alors, $(2a)^2 \equiv 1$ modulo 4 et la deuxième condition s'écrit $d(2b)^2 \equiv 1$ modulo 4. Elle n'a pas de solution dans ce cas lorsque $d \equiv 2, 3$ modulo 4 mais elle équivaut à $b \in \frac{1}{2}\mathbb{Z} \setminus \mathbb{Z}$ lorsque $d \equiv 1$ modulo 4. □

Unités d'un corps quadratique

Soit $K = \mathbb{Q}(\sqrt{d})$ un corps quadratique avec d sans facteur carré. Le groupe des unités de l'anneau $\mathbb{Z}_K$ est de nature différente suivant le signe de d. Si $d < 0$ alors ce groupe est fini et est

$$\begin{aligned} \{1, i, -1, -i\} \qquad & \text{si } d = -1 \\ \{1, -j^2, j, -1, j^2, -j\} \qquad & \text{si } d = -3 \\ \{1, -1\} \qquad & \text{si } d \leqslant -5. \end{aligned}$$

Si $d > 0$ alors ce groupe est infini. On appelle *unité fondamentale* de K la plus petite unité ε de $\mathbb{Z}_K$ qui soit strictement supérieure à 1. Le groupe des unités est engendré par -1 et ε.

Petit théorème de Fermat pour les corps quadratiques

2.2. — Théorème. *Soient $K = \mathbb{Q}(\sqrt{d})$ un corps quadratique, $\alpha \in \mathbb{Z}_K$ et p un nombre premier impair tel que $p \nmid \alpha$ dans l'anneau $\mathbb{Z}_K$. Alors,*

$$\alpha^p \equiv \begin{cases} \alpha & \text{si } (d/p) = 1 \\ \overline{\alpha} & \text{si } (d/p) = -1 \end{cases} \pmod{p}.$$

(Bien sûr, pour $u, v \in \mathbb{Z}_K$, l'identité $u \equiv v$ modulo p, signifie que $u - v$ appartient à l'idéal $p\mathbb{Z}_K$.)

Démonstration — Il existe $a, b \in \mathbb{Z}$ tels que $2\alpha = a + b\sqrt{d}$. Mais alors,

$$2\alpha^p \equiv (2\alpha)^p = (a + b\sqrt{d})^p \equiv a^p + b^p d^{(p-1)/2}\sqrt{d} \equiv a + b\left(\frac{d}{p}\right)\sqrt{d} \pmod{p}$$

et $a + b(d/p)\sqrt{d}$ vaut 2α ou $2\overline{\alpha}$ selon le signe de (d/p). □

Anneaux quadratiques, discriminant

2.3. — Définitions. Un *anneau quadratique* A est un Z-module libre de rang 2 et un sous-anneau (unitaire) de $\mathbb{C}$. Soit (α, β) une base du module A. Le *discriminant* de A est l'entier (non-nul)

$$\operatorname{disc}(A) = \det \begin{pmatrix} \operatorname{tr}(\alpha^2) & \operatorname{tr}(\alpha\beta) \\ \operatorname{tr}(\alpha\beta) & \operatorname{tr}(\beta^2) \end{pmatrix} = \left(\det \begin{pmatrix} \alpha & \overline{\alpha} \\ \beta & \overline{\beta} \end{pmatrix} \right)^2. \tag{1}$$

Si $D > 0$, on dit que A est un anneau quadratique *réel*. Si $D < 0$, on dit que A est un anneau quadratique *imaginaire*.

2.4. — Exercice. Les deux valeurs données pour la définition (1) du discriminant d'un anneau quadratique sont égales.

2.5. — Exercice. Soit A un anneau quadratique.

- Le module A admet une base de la forme $(1, \alpha)$ où α est un entier quadratique.
- Soit $K = \mathbb{Q}(\alpha)$ et $\mathbb{Z}_K = \mathbb{Z}[\omega]$ son anneau d'entiers. Il existe $f \in \mathbb{N}^*$ tel que $A = \mathbb{Z}[f\omega]$.

2.6. — Définition. Un *discriminant fondamental* est un entier non carré parfait vérifiant

$$\begin{cases} D \equiv 1 \text{ modulo } 4 \\ D \text{ sans facteur carré} \end{cases} \quad \text{ou} \quad \begin{cases} D \equiv 0 \text{ modulo } 4 \\ D/4 \equiv 2 \text{ ou } 3 \text{ modulo } 4 \text{ et sans facteur carré} \end{cases}$$

2.7. — Proposition. *Le discriminant de l'anneau des entiers d'un corps quadratique est un discriminant fondamental. Inversement, Si D est un discriminant fondamental, il existe un unique corps quadratique K tel que le discriminant de $\mathbb{Z}_K$ soit D.*

Démonstration — Exercice (utiliser 2.1). □

Classes d'idéaux

Soit d sans facteur carré et $K = \mathbb{Q}(\sqrt{d})$. L'anneau $\mathbb{Z}_K$ n'est que rarement principal. Lorsque $d < 0$, il n'est principal que pour neuf valeurs de d, à savoir

$$d = -1, -2, -3, -7, -11, -19, -43, -67, -163.$$

Lorsque $d > 0$, Gauss a conjecturé que l'anneau $\mathbb{Z}_K$ est principal pour un nombre infini de valeurs de d mais cela n'a jamais été démontré.

2.8. — **Définition.** Soit A un anneau quadratique, et K le corps quadratique contenant A. Un idéal I de A est dit *propre* (à A) si le stabilisateur de I dans K, c'est-à-dire l'ensemble

$$\{\lambda \in K \mid \lambda I \subseteq I\}$$

est exactement égal à A. Autrement dit, A est le plus grand sous-anneau de K dont I soit un idéal.

2.9. — **Définitions.** Soient I, J deux idéaux (non nuls) d'un anneau quadratique A. On dit que les idéaux I, J sont *équivalents* s'il existe $a, b \in A$ tels que $aI = bJ$. Si de plus les normes de a et b sont positives, on dit que les idéaux I, J sont *strictement équivalents*.

La multiplication des idéaux induit une loi de groupe sur l'ensemble des classes d'idéaux propres de A pour la relation d'équivalence ci-dessus. Le groupe obtenu est noté $H(A)$. On obtient de la même façon une structure de groupe sur l'ensemble des classes d'idéaux propres de A pour la relation d'équivalence stricte ci-dessus. Ce groupe est noté $H^+(A)$. Ces deux groupes sont finis et leurs ordres respectifs sont notés $h(A)$ et $h^+(A)$. L'anneau A est principal si et seulement si $h(A)$ vaut 1. Lorsque A est l'anneau des entiers d'un corps quadratique K, on utilise aussi les notations $H(K)$, $H^+(K)$, $h(K)$, $h^+(K)$.

2.10. — **Remarque.** Si l'anneau quadratique est imaginaire, les normes d'éléments sont positives, donc les notions d'équivalence et d'équivalence stricte coïncident, et les groupes $H(A)$ et $H^+(A)$ aussi. Si l'anneau quadratique est réel, la situation dépend du signe de la norme de l'unité fondamentale. Si ce signe est négatif alors les notions d'équivalence et d'équivalence stricte coïncident encore (parce qu'il suffit de multiplier a et b par l'unité fondamentale, si leur norme est négative, pour la rendre positive). Sinon, les classes au sens large se scindent chacune en deux classes au sens strict.

3. Formes quadratiques binaires

De bonnes références pour les formes quadratiques sont [33], [34] et (avec application à la factorisation) [119].

3.1. — **Définitions.** Une *forme quadratique binaire* est un polynôme $q(x, y) = ax^2 + bxy + cy^2 \in \mathbb{Z}[x, y]$ tel que $D = b^2 - 4ac$ ne soit pas un carré parfait. L'entier D est appelé le *discriminant* de q. Si $D > 0$, on parle de forme quadratique *réelle*. Si $D < 0$, on parle de forme quadratique *imaginaire*, et on ne s'intéresse dans ce cas qu'aux formes définies positives (i.e. telles que $a > 0$).

3.2. — **Remarque.** Un discriminant de forme quadratique est donc un entier non carré parfait et congru à 0 ou 1 modulo 4. Inversement, tout non carré parfait congru à 0 ou 1 modulo 4 est un discriminant de forme quadratique (par exemple $x^2 - (D/4)y^2$ si $D \equiv 0$ modulo 4, et $x^2 + xy + ((1 - D)/4)y^2$ si $D \equiv 1$ modulo 4.

3.3. — **Définition.** Une forme quadratique $ax^2 + bxy + cy^2$ est dite *primitive* si $\mathrm{pgcd}(a, b, c) = 1$.

3.4. — **Exercice.** Soit D un discriminant (voir remarque 3.2).

• D est fondamental (définition 2.6) si et seulement si il n'existe pas de discriminant de la forme D/f^2 avec $f > 1$.

• D est fondamental si et seulement si toutes les formes quadratiques de discriminant D sont primitives.

Equivalence et lien avec les classes d'idéaux

On note $\mathrm{SL}_2(\mathbb{Z})$ l'ensemble des matrices 2×2 à coefficients entiers et de déterminant 1.

3.5. — Définition. On dit que deux formes quadratiques $q(x,y)$ et $q'(x,y)$ sont équivalentes s'il existe une matrice $\left(\begin{smallmatrix} p & r \\ s & t \end{smallmatrix}\right) \in \mathrm{SL}_2(\mathbb{Z})$ telle que $q(px+ry, sx+ty) = q'(x,y)$.

3.6. — Exercice. • Deux formes quadratiques équivalentes ont même discriminant.
• Si deux formes quadratiques sont équivalentes et si l'une d'elle est primitive alors l'autre aussi.

On note $H(D)$ l'ensemble des classes de formes quadratiques primitives (et définies positives si $D<0$) de discriminant D.

3.7. — Définition. Soient K un corps quadratique de discriminant D et u, v deux éléments de K tels que $u\overline{v} \neq \overline{u}v$. Le couple (u,v) est dit *orienté* (positivement) si l'on a la relation

$$u\overline{v} - \overline{u}v > 0 \quad \text{si } D>0, \qquad\qquad \mathrm{Im}(u/v) > 0 \quad \text{si } D<0. \tag{2}$$

Soit A un anneau quadratique de discriminant D. Les idéaux de A sont des $\mathbb{Z}$-modules libres de rang 2. Lorsque u, v forment une Z-base d'un idéal I, on note $I = [u,v]$. Tout idéal de A peut s'écrire sous cette forme, où (u,v) est un couple orienté. On peut alors montrer que l'on définit deux applications bijectives entre $H^+(A)$ et $H(D)$ par les formules suivantes.

$$\mathcal{F} : \begin{cases} H^+(A) \longrightarrow H(D) \\ \qquad I \longmapsto N(vx-uy)/N(I) \end{cases} \tag{3}$$

où $[u,v]$ est une base orientée de I, et

$$\mathcal{I} : \begin{cases} H(D) \longrightarrow H^+(A) \\ \qquad q \longmapsto [u,v] \end{cases}$$

où (u,v) est un couple orienté d'éléments non nuls de A vérifiant $q(u,v) = 0$ et $\operatorname{sgn} N(v) = \operatorname{sgn}(a)$. Cela permet d'établir le résultat suivant :

3.8. — Théorème. Soit A un anneau quadratique de discriminant D. Les ensembles $H(D)$ et $H^+(A)$ sont en bijection. □

Réduction

3.9. — Définitions. • Une forme quadratique imaginaire (non nécessairement primitive) $ax^2+bxy+cy^2$ de discriminant D est dite *réduite* si

$$|b| \leqslant a \leqslant c \qquad \text{et en plus } b>0 \text{ si l'une des inégalités n'est pas stricte.}$$

• Une forme quadratique réelle primitive $ax^2+bxy+cy^2$ de discriminant D est dite *réduite* si

$$\left|\sqrt{D} - 2|a|\right| < b < \sqrt{D}.$$

3.10. — Proposition. *Soit $ax^2+bxy+cy^2$ une forme quadratique imaginaire réduite de discriminant D. On a la relation*

$$a \leqslant \sqrt{|D|/3}.$$

Démonstration — Parce que $4a^2 \leqslant 4ac = b^2 + |D| \leqslant 4a^2 + |D|$, on a $3a^2 \leqslant |D|$. □

3.11. — **Exercice.** Enumérer les formes quadratiques réduites de discriminant -39.

3.12. — **Proposition.** Toute forme quadratique est équivalente à une forme quadratique réduite. Dans le cas imaginaire, cette forme quadratique réduite est unique.

Démonstration — Exercice. Il s'agit d'appliquer les transformations définies par les matrices $S = \left(\begin{smallmatrix} 0 & -1 \\ 1 & 0 \end{smallmatrix}\right)$ et $T_\lambda = \left(\begin{smallmatrix} 1 & \lambda \\ 0 & 1 \end{smallmatrix}\right)$:

$$S(ax^2+bxy+cy^2) = cy^2 - bxy + ay^2, \qquad T_\lambda(ax^2+bxy+cy^2) = ax^2 + (b+2\lambda a)xy + (\lambda^2 a + \lambda b + c)y^2,$$

selon l'un des algorithmes suivants. □

3.13. — **Algorithme.** On peut réduire une forme quadratique imaginaire par l'algorithme suivant :

Entrée : Une forme quadratique imaginaire $q(x,y) = ax^2+bxy+cy^2$ de discriminant D.
Sortie : Une forme quadratique réduite équivalente à q.

Répéter
  Appliquer une transformation T_λ de telle sorte que $-a < b \leqslant a$
  Si q est réduite alors retourner q et arrêter l'algorithme
  Appliquer une transformation S
Fin répéter

3.14. — **Exemple.** Réduisons la forme quadratique $47x^2+33xy+6y^2$ de discriminant -39 :

$$47x^2+33xy+6y^2 \overset{S}{\longmapsto} 6x^2-33xy+47y^2 \overset{T_3}{\longmapsto} 6x^2+3xy+2y^2 \overset{S}{\longmapsto} 2x^2-3xy+6y^2 \overset{T_1}{\longmapsto} 2x^2+xy+5y^2.$$

3.15. — **Algorithme.** On peut réduire une forme quadratique réelle (éventuellement déjà réduite) par l'algorithme suivant :

Entrée : Une forme quadratique réelle $q(x,y) = ax^2+bxy+cy^2$ de discriminant D.
Sortie : Une forme quadratique réduite équivalente à q.

Répéter
  Appliquer une transformation S
  Si q est réduite alors retourner q et arrêter l'algorithme
  Si $|a| > \sqrt{D}$ alors appliquer une transformation T_λ de telle sorte que $-|a| < b < |a|$
  Sinon appliquer une transformation T_λ de telle sorte que $\sqrt{D} - 2|a| < b < \sqrt{D}$
Fin répéter

3.16. — **Remarque.** Dans le cas des formes quadratiques réelle, il y a en général plusieurs formes réduites dans la même classe d'équivalence, et l'algorithme de réduction permute ces formes réduites selon un cycle.

3.17. — **Exemple.** Calculons les formes réduites équivalentes à la forme $-13x^2-8xy+3y^2$ de discriminant 220. Les formes réduites sont marquées par un $(*)$:

$$\begin{aligned} -13x^2-8xy+3y^2 &\overset{S}{\longmapsto} 3x^2+8xy-13y^2 \overset{T_{-1}}{\longmapsto} 3x^2+2xy-18y^2 \overset{T_4}{\longmapsto} 3x^2+14xy-2y^2(*) \\ &\overset{S}{\longmapsto} -2x^2-14xy+3y^2 \overset{T_7}{\longmapsto} -2x^2+14xy+3y^2(*) \overset{S}{\longmapsto} 3x^2-14xy-2y^2 \\ &\overset{T_4}{\longmapsto} 3x^2+10xy-10y^2(*) \overset{S}{\longmapsto} -10x^2-10xy+3y^2 \overset{T_1}{\longmapsto} -10x^2+10xy+3y^2(*) \\ &\overset{S}{\longmapsto} 3x^2-10xy-10y^2 \overset{T_4}{\longmapsto} 3x^2+14xy-2y^2(*). \end{aligned}$$

3.18. — **Exemple.** Pour $D = 316$ le cycle de la forme principale est le suivant :

$$x^2+16xy-15y^2 \longmapsto -15x^2+14xy+2y^2 \longmapsto 2x^2+14xy-15y^2 \longmapsto -15x^2+16xy+y^2 \longmapsto x^2+16xy-15y^2.$$

Représentation

3.19. — Définitions. On dit qu'une forme quadratique $q(x,y)$ *représente* un entier m s'il existe des entiers u, v tels que $q(u,v) = m$. Si de plus u et v sont premiers entre eux, on dit que q *représente proprement* m.

3.20. — Lemme. *Soient q une forme quadratique et m un entier. Alors q représente proprement m si et seulement si elle est équivalente à une forme dont le coefficient en x^2 est m.*

Démonstration — Supposons que q soit équivalente à $mx^2 + bxy + cy^2$. Alors, il existe $\left(\begin{smallmatrix} p & r \\ s & t \end{smallmatrix}\right) \in \mathrm{SL}_2(\mathbb{Z})$ telle que

$$q(px + ry, sx + ty) = mx^2 + bxy + cy^2.$$

Pour $x = 1$ et $y = 0$, on obtient $q(p,s) = m$. De plus, p et s sont premiers entre eux puisqu'ils vérifient l'identité de Bézout.

Inversement, supposons qu'il existe des entiers p, s premiers entre eux tels que $q(p,s) = m$. D'après Bézout, il existe r, t tels que $pt - rs = 1$. Le coefficient en x^2 de la forme $q(px + ry, sx + ty)$ est m. □

3.21. — Lemme. *Soit q une forme primitive et m un entier non nul. Alors q représente proprement un entier étranger à m.*

Démonstration — Notons d'abord, puisque q est primitive, que les trois entiers

$$q(1,0) = a, \qquad q(0,1) = c, \qquad q(1,1) = a + b + c$$

sont globalement premiers entre eux. Donc, pour chaque p premier diviseur de m, on peut trouver $(x_p, y_p) \in \{(1,0),(0,1),(1,1)\}$ tels que $q(x_p, y_p)$ ne soit pas divisible par p. Enfin, le théorème chinois assure l'existence d'entiers x, y tels que $x \equiv x_p$ et $y \equiv y_p$ modulo p pour chaque p premier divisant m. Alors $q(x,y)$ est étranger à m. De plus, si d désigne $\gcd(x,y)$, l'entier $q(x/d, y/d)$ est aussi étranger à m et est proprement représenté. □

Loi de composition

La loi de groupe sur $H^+(A)$ se transporte sur $H(D)$. On retrouve alors la loi de composition sur les formes quadratiques découverte par Gauss :

3.22. — Loi de composition. *Soit D un discriminant. La loi de composition induite sur $H(D)$ par $H^+(D)$ s'exprime de la manière suivante. Soient $q_1 = a_1x^2 + b_1xy + c_1y^2$ et $q_2 = a_2x^2 + b_2xy + c_2y^2$ deux formes quadratiques primitives de discriminant D. D'après les lemmes 3.20 et 3.21, on peut supposer que a_1 et a_2 sont premiers entre eux. Il existe donc des entiers h_1, h_2 tels que*

$$a_1h_1 - a_2h_2 = \frac{b_2 - b_1}{2}. \tag{4}$$

La composition de la classe de q_1 par la classe de q_2 est la classe de la forme quadratique suivante

$$Q = a_1a_2x^2 + Bxy + Cy^2 \qquad \text{où} \qquad B = b_1 + 2a_1h_1 = b_2 + 2a_2h_2 \quad \text{et} \quad C = \frac{B^2 - D}{4a_1a_2}.$$

4. Fractions continues

On note

$$[a_0, a_1, a_2, \cdots, a_n] = a_0 + \cfrac{1}{a_1 + \cfrac{1}{\ddots \qquad \vdots \\ a_{n-2} + \cfrac{1}{a_{n-1} + \cfrac{1}{a_n}}}}.$$

Pour $x \in \mathbb{R}$, on pose $a_0 = \lfloor x \rfloor$, $x_0 = 1/(x - a_0)$, puis $a_i = \lfloor x_{i-1} \rfloor$ et $x_i = 1/(x_{i-1} - a_i)$, pour $i = 1, 2, \ldots$, (tant que ces expressions ont un sens). On a donc

$$x = [a_0, a_1, \ldots, a_n, x_n] \qquad \text{pour tout } n.$$

La suite (finie ou infinie) des a_i s'appelle le *développement en fraction continue de x*. Les nombres rationnels $p_n/q_n = [a_0, \ldots, a_n]$ s'appellent les *quotients partiels de x*. Les x_i sont les *restes partiels de x*.

Si le développement de x en fraction continue est fini alors x est rationnel. Inversement, si $x = p/q$ est rationnel, alors les quotients partiels s'obtiennent en appliquant l'algorithme d'Euclide à p et q et le développement de x en fraction continue est fini.

Fractions continues de nombres quadratiques

Si x est un nombre algébrique de degré 2 sur $\mathbb{Q}$, il s'écrit sous la forme $x = (\sqrt{d} - u)/v$ avec, $u, v, d \in \mathbb{Z}$ et $v \neq 0$. On peut même supposer que v divise $u^2 - d$ puisque sinon $x = (\sqrt{dv^2} - u|v|)/v|v|$. Sous ces conditions, on peut calculer les termes (a_n) et restes partiels (x_n) successifs de x par les formules suivantes. On pose d'abord

$$a_0 = \lfloor x \rfloor, \qquad v_0 = v, \qquad u_0 = u + a_0 v_0,$$

de telle sorte que $x = a_0 + 1/x_0$ et $1/x_0 = (\sqrt{d} - u_0)/v_0$. Puis on calcule ensuite, pour $n \in \mathbb{N}$,

$$v_{n+1} = \frac{d - u_n^2}{v_n}, \qquad a_{n+1} = \left\lfloor \frac{\sqrt{d} + u_n}{v_{n+1}} \right\rfloor, \qquad u_{n+1} = a_{n+1} v_{n+1} - u_n,$$

de telle sorte que les restes partiels vérifient $1/x_n = (\sqrt{d} - u_n)/v_n = v_{n+1}/(\sqrt{d} + u_n)$. On notera aussi que la division dans le calcul de v_{n+1} est toujours exacte. Knuth [56] préconise même de remplacer les formules ci-dessus par

$$v_{n+1} = a_n(u_{n-1} - u_n) + v_{n-1}, \qquad a_{n+1} = \begin{cases} \left\lfloor \dfrac{\lfloor \sqrt{d} \rfloor + u_n}{v_{n+1}} \right\rfloor & \text{si } v_{n+1} > 0 \\ \left\lfloor \dfrac{\lfloor \sqrt{d} \rfloor + u_n + 1}{v_{n+1}} \right\rfloor & \text{si } v_{n+1} < 0 \end{cases}$$

pour le calcul de v_{n+1} et a_{n+1}. La première d'entre elles résulte de

$$(u_{n-1} + u_n)(v_{n+1} - v_{n-1}) = a_n v_n (v_{n+1} - v_{n-1}) = a_n\big((d - u_n^2) - (d - u_{n-1}^2)\big) = a_n(u_{n-1}^2 - u_n^2).$$

Polynômes continuants

On peut définir les *polynômes continuants* $(Q_n)_{n \geqslant -1} \in \mathbb{Z}[A_i \mid i \in \mathbb{N}]$ de la manière suivante :

$$\begin{aligned} Q_{-1} = 0, \qquad Q_0 = 1, \\ Q_{n+1}(A_1, \ldots, A_{n+1}) = A_{n+1} Q_n(A_1, \ldots, A_n) + Q_{n-1}(A_1, \ldots, A_{n-1}) \qquad \text{pour } n \geqslant 0. \end{aligned} \tag{5}$$

Chaque Q_n est donc un polynôme à n variables et les termes suivants sont

$$\begin{gathered} Q_1(A_1) = A_1, \quad Q_2(A_1, A_2) = A_1 A_2 + 1, \quad Q_3(A_1, A_2, A_3) = A_1 A_2 A_3 + A_1 + A_3, \\ Q_4(A_1, A_2, A_3, A_4) = A_1 A_2 A_3 A_4 + A_1 A_2 + A_1 A_4 + A_3 A_4 + 1, \\ Q_5(A_1, A_2, A_3, A_4, A_5) = A_1 A_2 A_3 A_4 A_5 + A_1 A_2 A_3 + A_1 A_2 A_5 + A_1 A_4 A_5 + A_3 A_4 A_5 + A_1 + A_3 + A_5. \end{gathered}$$

Le polynôme Q_n est la somme des monômes obtenus en partant de $A_1 A_2 \cdots A_n$ et en supprimant un nombre quelconque de paires adjacentes $A_k A_{k+1}$. On a donc $Q_n(A_1, \ldots, A_n) = Q_n(A_n, \ldots, A_1)$ et par cette symétrie, la relation récurrente de la définition devient

$$Q_n(A_1, \ldots, A_n) = A_1 Q_{n-1}(A_2, \ldots, A_n) + Q_{n-2}(A_3, \ldots, A_n). \tag{6}$$

4.1. — Proposition. *Pour $n \in \mathbb{N}$, on a*

$$[A_0, \dots, A_n] = \frac{Q_{n+1}(A_0, \dots, A_n)}{Q_n(A_1, \dots, A_n)}$$

et, même lorsque l'on spécialise les A_i par des valeurs de $\mathbb{N}^*$, la fraction du membre de droite est réduite.

Démonstration — La relation est claire pour $n = 0$. Nous supposons qu'elle est vraie au rang $n-1$ et donc que

$$[A_1, \dots, A_n] = \frac{Q_n(A_1, \dots, A_n)}{Q_{n-1}(A_2, \dots, A_n)}.$$

Alors on a

$$\begin{aligned}[A_0, \dots, A_n] &= A_0 + \frac{1}{[A_1, \dots, A_n]} = A_0 + \frac{Q_{n-1}(A_2, \dots, A_n)}{Q_n(A_1, \dots, A_n)} \\ &= \frac{A_0 Q_n(A_1, \dots, A_n) + Q_{n-1}(A_2, \dots, A_n)}{Q_n(A_1, \dots, A_n)} = \frac{Q_{n+1}(A_0, \dots, A_n)}{Q_n(A_1, \dots, A_n)}.\end{aligned}$$

On obtient donc la relation pour chaque n par récurrence. Le fait que les fractions soient réduites s'obtient facilement, par récurrence aussi, en utilisant encore (6). □

4.2. — Corollaire. *Soient x un nombre réel et $a_0, \dots, a_n$ le début de son développement en fraction continue. Alors le n-ième quotient partiel p_n/q_n de x est donné en fraction réduite par*

$$p_n = Q_{n+1}(a_0, \dots, a_n) \qquad \text{et} \qquad q_n = Q_n(a_1, \dots, a_n).$$

Démonstration — Immédiat. □

4.3. — Corollaire. *Soient x un nombre réel, (a_n) le développement de x en fraction continue et (p_n/q_n) ses quotients partiels sous forme réduite. Alors on a les relations (lorsque les indices leur donnent un sens)*

$$p_{n+1} = a_{n+1} p_n + p_{n-1} \qquad \text{et} \qquad q_{n+1} = a_{n+1} q_n + q_{n-1}.$$

Démonstration — C'est une reformulation de (5). □

4.4. — Proposition. *Pour $n \in \mathbb{N}$, on a la relation*

$$Q_n(A_0, \dots, A_{n-1}) Q_n(A_1, \dots, A_n) - Q_{n-1}(A_1, \dots, A_{n-1}) Q_{n+1}(A_0, \dots, A_n) = (-1)^n. \tag{7}$$

Démonstration — La relation est claire pour $n = 0$. Désignons par I_n le membre de gauche de l'égalité à montrer. Nous avons

$$\begin{aligned}I_{n+1} &= Q_{n+1}(A_0, \dots, A_n) Q_{n+1}(A_1, \dots, A_{n+1}) - Q_n(A_1, \dots, A_n) Q_{n+2}(A_0, \dots, A_{n+1}) \\ &= Q_{n+1}(A_0, \dots, A_n) Q_{n+1}(A_1, \dots, A_{n+1}) - Q_n(A_1, \dots, A_n)\big(A_{n+1} Q_{n+1}(A_0, \dots, A_n) + Q_n(A_0, \dots, A_{n-1})\big) \\ &= Q_{n+1}(A_0, \dots, A_n)\big(Q_{n+1}(A_1, \dots, A_{n+1}) - A_{n+1} Q_n(A_1, \dots, A_n)\big) - Q_n(A_0, \dots, A_{n-1}) Q_n(A_1, \dots, A_n) \\ &= Q_{n+1}(A_0, \dots, A_n) Q_{n-1}(A_1, \dots, A_{n-1}) - Q_n(A_0, \dots, A_{n-1}) Q_n(A_1, \dots, A_n) = -I_n\end{aligned}$$

On a donc le résultat par récurrence. □

4.5. — Corollaire. *Soient x un nombre réel, p_{n-1}/q_{n-1} et p_n/q_n deux quotients partiels successifs (on suppose que ces quotients sont définis). Alors on a la relation*

$$p_{n-1} q_n - p_n q_{n-1} = (-1)^n.$$

Démonstration — Immédiat. □

Approximation par des quotients partiels

4.6. — Théorème. *Soit $x \in \mathbb{R}$. Les quotients partiels p_n/q_n de x vérifient (on suppose que le $n+1$-ème quotient partiel de x existe) :*

$$\left| x - \frac{p_n}{q_n} \right| \leqslant \frac{1}{q_n q_{n+1}} < \frac{1}{q_n^2}.$$

DÉMONSTRATION — Désignons par $a_0, \dots, a_n$ le début du développement en fraction continue de x. On a

$$\begin{aligned} x - p_n/q_n &= \frac{Q_{n+2}(a_0,\dots,a_n,x_n)}{Q_{n+1}(a_1,\dots,a_n,x_n)} - \frac{Q_{n+1}(a_0,\dots,a_n)}{Q_n(a_1,\dots,a_n)} \\ &= \frac{Q_{n+2}(a_0,\dots,a_n,x_n)Q_n(a_1,\dots,a_n) - Q_{n+1}(a_0,\dots,a_n)Q_{n+1}(a_1,\dots,a_n,x_n)}{Q_n(a_1,\dots,a_n)Q_{n+1}(a_1,\dots,a_n,x_n)} \\ &= \frac{(-1)^n}{Q_n(a_1,\dots,a_n)Q_{n+1}(a_1,\dots,a_n,x_n)}. \end{aligned}$$

Mais on a $x_n \geqslant a_{n+1}$. Donc

$$\begin{gathered} Q_n(a_1,\dots,a_n) = q_n \\ Q_{n+1}(a_1,\dots,a_n,x_n) \geqslant Q_{n+1}(a_1,\dots,a_n,a_{n+1}) = q_{n+1} \end{gathered}$$

d'où le résultat. □

On a la réciproque partielle et très importante suivante.

4.7. — Théorème. *Soient x un réel et p/q un rationnel vérifiant $|x - p/q| < 1/2q^2$. Alors la fraction p/q est l'un des quotients partiels du développement de x.*

DÉMONSTRATION — Voir, par exemple, dans [70]. □

Fractions continues de nombres quadratiques (suite)

4.8. — Théorème. *Soit $x \in \mathbb{R} \setminus \mathbb{Q}$. Le développement de x en fraction continue est ultimement périodique si et seulement si x est algébrique quadratique.*

DÉMONSTRATION — Supposons que le développement de x soit ultimement périodique. Il existe donc $n, k \in \mathbb{N}^*$ tels que $x_n = x_{n+k}$. On a alors

$$[a_0,\dots,a_n,x_n] = [a_0,\dots,a_{n+k},x_n]$$

et donc

$$x_n = [a_{n+1},\dots,a_{n+k},x_n] = \phi_{n+1} \circ \cdots \circ \phi_{n+k}(x_n)$$

où ϕ_i est l'application homographique $t \mapsto a_i + 1/t$. La composée des ϕ_i est donc une homographie dont x_n est point fixe. On en déduit que x_n est algébrique quadratique et que x aussi.

Réciproquement, supposons qu'il existe des entiers A, B, C tels que $Ax^2 + Bx + C = 0$. Pour tout $n \in \mathbb{N}$ on a

$$x = [a_0,\dots,a_n,x_n] = \frac{Q_{n+2}(a_0,\dots,a_n,x_n)}{Q_{n+1}(a_1,\dots,a_n,x_n)} = \frac{x_n Q_{n+1}(a_0,\dots,a_n) + Q_n(a_0,\dots,a_{n-1})}{x_n Q_n(a_1,\dots,a_n) + Q_{n-1}(a_1,\dots,a_{n-1})} = \frac{x_n p_n + p_{n-1}}{x_n q_n + q_{n-1}}.$$

En posant $f(u,v) = Au^2 + Buv + Cv^2$. On en déduit que $f(x_n p_n + p_{n-1}, x_n q_n + q_{n-1}) = 0$ ou encore $A_n x_n^2 + B_n x_n + C_n = 0$ avec

$$A_n = f(p_n, q_n), \qquad C_n = f(p_{n-1}, q_{n-1}), \qquad B_n = f(p_n + p_{n-1}, q_n + q_{n-1}) - A_n - C_n.$$

De plus, il y a égalité des discriminants $B_n^2 - 4A_nC_n$ et $B^2 - 4AC$. D'après le théorème 4.6, il existe ε_n tel que $|\varepsilon| < 1$ et $p_n = xq_n + \varepsilon_n/q_n$. On obtient alors

$$\begin{aligned} A_n = f(p_n, q_n) &= (Ax^2 + Bx + c)q_n^2 + 2Ax\varepsilon_n + B\varepsilon_n + A\varepsilon_n^2/q_n^2 \\ &= 2Ax\varepsilon_n + B\varepsilon_n + A\varepsilon_n^2/q_n^2 \end{aligned}.$$

et donc $|A_n| \leqslant 2|xA| + |A| + |B|$. C_n et borné de la même façon et B_n est aussi borné puisque $B_n^2 = B^2 - 4AC + 4A_nC_n$. Le nombre de triplets (A_n, B_n, C_n) possibles est fini et donc le nombre de restes partiels x_n aussi. Le développement de x est ultimement périodique. □

5. Suites de Lucas

Le livre de Ribenboim [111] consacre de nombreuses pages aux suites de Lucas.

Soit P, Q deux entiers tels que $D = P^2 - 4Q$ ne soit pas un carré parfait de telle sorte que le polynôme unitaire $X^2 - PX + Q$ soit irréductible sur $\mathbb{Q}$. Notons α, β ses deux racines dans $\mathbb{Q}(\sqrt{D})$. Les *suites de Lucas de paramètres* P, Q sont les deux suites d'entiers $(U_n)_{n\in\mathbb{N}}$ et $(V_n)_{n\in\mathbb{N}}$ définies par

$$\begin{cases} U_0 = 0, & U_1 = 1, \\ V_0 = 2, & V_1 = P, \end{cases} \quad \text{et, pour } k \geqslant 0, \quad \begin{cases} U_{k+2} = PU_{k+1} - QU_k, \\ V_{k+2} = PV_{k+1} - QV_k. \end{cases}$$

Une récurrence facile montre qu'on a en fait

$$U_n = \frac{\alpha^n - \beta^n}{\alpha - \beta}, \qquad V_n = \alpha^n + \beta^n, \qquad \text{pour tout } n \in \mathbb{N}. \tag{8}$$

Il en résulte que la donnée du couple (U_n, V_n) est équivalente à la donnée de α^n.

On peut montrer de nombreuses relations sur les suites U et V. Parmi elles, les suivantes permettent en utilisant un algorithme judicieux, de calculer U_n et V_n pour de grandes valeurs de n.

$$\begin{cases} U_{2k} = U_k V_k \\ V_{2k} = V_k^2 - 2Q^k \end{cases} \qquad \begin{cases} 2U_{k+1} = PU_k + V_k \\ 2V_{k+1} = DU_k + PV_k \end{cases} \qquad \text{pour tout } k \in \mathbb{N}.$$

Propriétés liées aux nombres premiers

Le théorème suivant est l'analogue du petit théorème de Fermat pour les suites de Lucas. Comme plus haut, on désigne par P, Q des entiers tels que $D = P^2 - 4Q$ ne soit pas un carré parfait, et α, β sont les racines du polynôme $X^2 - PX + Q$.

5.1. — **Théorème.** *Soit p un nombre premier impair et désignons par ε le symbole de Jacobi (D/p). Si $\varepsilon = 1$, supposons de plus que p ne divise pas Q. On a la relation :*

$$p \mid U_{p-\varepsilon}.$$

Démonstration — • Si $\varepsilon = 1$ on a, d'après le théorème 2.2, $\alpha^p \equiv \alpha$ et $\beta^p \equiv \beta$ modulo p. Mais puisque $p \nmid Q = \alpha\beta$, les entiers algébriques α et β sont inversibles modulo $p\mathbb{Z}_K$. On obtient donc $\alpha^{p-1} \equiv 1 \equiv \beta^{p-1}$ modulo $p\mathbb{Z}_k$. Comme $\alpha - \beta = \sqrt{D}$ est aussi inversible modulo $p\mathbb{Z}_K$ (car $\varepsilon \neq 0$), on en déduit que U_{p-1} est dans l'idéal $p\mathbb{Z}_K$, d'après (8). Mais U_{p-1} est un entier rationnel donc $p \mid U_{p-1}$ dans l'anneau $\mathbb{Z}$.

• Si $\varepsilon = -1$ on a, d'après le théorème 2.2, $\alpha^p \equiv \beta$ et $\beta^p \equiv \alpha$ modulo p et donc $\alpha^{p+1} \equiv \alpha\beta \equiv \beta^{p+1}$ modulo $p\mathbb{Z}_K$. Donc $p \mid U_{p+1}$ (dans $\mathbb{Z}_K$ donc dans $\mathbb{Z}$ puisqu'ils sont tous deux dans $\mathbb{Z}$) d'après (8).

• Si $p \mid D$ alors $U_p = \alpha^{p-1} + \alpha^{p-1}\beta + \cdots \beta^{p-1} \equiv p\alpha^{p-1}$ modulo $\sqrt{D}$ puisque $\alpha \equiv \beta$ modulo $\sqrt{D}$. Donc il existe $\lambda \in \mathbb{Z}_K$ tel que $U_p = p\alpha^{p-1} + \lambda\sqrt{D}$. Mais alors $U_p^2 = p\alpha^{p-1}(p\alpha^{p-1} + 2\lambda\sqrt{D}) + \lambda^2 D \equiv 0$ modulo p. Puisque p et U_p sont dans $\mathbb{Z}$, on a $p \mid U_p^2$ dans $\mathbb{Z}$ et donc $p \mid U_p$ puisque p est premier (attention au raisonnement : p est irréductible dans $\mathbb{Z}$ mais peut-être pas dans $\mathbb{Z}_K$). □

Comme on l'a fait pour le petit théorème de Fermat par la version forte 5.1.22, on peut renforcer le théorème précédent. On obtient le résultat ci-dessous, qui fait intervenir les suites (V_n).

5.2. — **Théorème.** *Soit p un premier ne divisant pas $2QD$ et posons $\varepsilon = (D/p)$ et $p - \varepsilon = 2^k q$ avec q impair. Alors l'une ou l'autre des conditions suivantes est vérifiée :*

$$p \mid U_q \qquad \textit{ou} \qquad \textit{il existe un } i \textit{ tel que } 0 \leqslant i < k \textit{ et } p \mid V_{2^i q}$$

Démonstration — La démonstration est semblable à celle du théorème 5.1.22, en tenant compte de l'identité $U_{2j} = U_j V_j$. Considérons le plus petit entier i tel que $p \mid U_{2^i q}$ (il existe, d'après 5.1). Si $i > 0$ alors $p \mid V_{2^{i-1}q}$ puisque $U_{2^i q} = U_{2^{i-1}q} V_{2^{i-1}q}$ et $p \nmid U_{2^{i-1}q}$. □

6. Courbes elliptiques

La théorie des courbes elliptiques est vaste et difficile. Des références complètes sont les livres de Silverman [123] et [124], de Husemöller [52] et Koblitz [58]. L'article de Cohen [31] est une excellente introduction liée à la cryptographie. Le livre de Menezes [82] est une référence adaptée aux usages cryptographiques. Les démonstrations de la plupart des résultats rassemblés ici sont hors de portée de ce cours.

Soit K un corps, on appelle *courbe elliptique sur* K une courbe dans le plan projectif $\mathbb{P}^2(K)$, cubique et sans points singuliers, et munie d'un point distingué qui jouera un rôle particulier (il sera l'élément neutre dans le groupe défini plus loin). Elle est donc définie par un polynôme irréductible homogène en trois variables à coefficients dans K (et par la donnée du point distingué). Par un changement de variables homographique, on peut toujours se ramener à une équation dite *de Weierstrass* (forme longue) :

$$Y^2Z + a_1XYZ + a_3YZ^2 = X^3 + a_2X^2Z + a_4XZ^2 + a_6Z^3 \qquad \text{où les } a_i \in K. \tag{9}$$

En coordonnées affines (c'est-à-dire en posant $X = x/z$, $Y = y/z$ et $Z = 1$) on a donc

$$y^2 + a_1xy + a_3y = x^3 + a_2x^2 + a_4x + a_6.$$

En caractéristique autre que 2 ou 3, on peut toujours trouver une "forme courte" de Weierstrass, c'est-à-dire mettre E sous la forme

$$Y^2Z = X^3 + aXZ^2 + bZ^3 \quad \text{(coordonnées projectives)}, \qquad y^2 = x^3 + ax + b \quad \text{(coordonnées affines)}. \tag{10}$$

La courbe définie par (10) admet un unique point *à l'infini* (i.e. avec $Z = 0$), de coordonnées $(0:1:0)$. C'est en général ce point qui sera distingué. On appelle discriminant de cette courbe l'élément $-16(4a^3 + 27b^2)$ de K. Le facteur entre parenthèses est le discriminant du polynôme membre de droite.

6.1. — Lemme. *Une courbe définie par une équation (10) est non-singulière si et seulement si son discriminant est non nul.*

Démonstration — Soit $F(X, Y, Z) = X^3 + aXZ^2 + bZ^3 - Y^2Z$. Ses trois dérivées partielles sont $F_X = 3X^2 + aZ^2$, $F_Y = -2YZ$ et $F_Z = 2aXZ + 3bZ^2 - Y^2$. La courbe est singulière si et seulement si F et ses trois dérivées partielles ont un zéro commun $(X : Y : Z)$ dans une extension. Dans le cas $Z = 0$, on obtient facilement $X = Y = 0$, ce qui ne correspond pas à un point projectif. On peut donc supposer $Z = 1$, et on a alors $Y = 0$ puis $X^3 + aX + b = 3X^2 + a = 0$. On voit alors qu'un point singulier existe si et seulement s'il existe $X \in K$ tel que $a = -3X^2$ et $b = 2X^3$. Ceci est équivalent à la condition $4a^3 + 27b^2 = 0$. □

Cas des caractéristiques 2 et 3

En caractéristique 3, on ne peut pas simplifier autant la forme de Weierstrass. On peut tout de même toujours trouver une équation affine de la forme

$$y^2 = x^3 + ax^2 + bx + c \qquad a, b, c \in K.$$

En caractéristique 2 on peut se ramener à une équation de l'une des deux formes suivantes.

$$y^2 + xy = x^3 + ax^2 + b \quad \text{ou} \quad y^2 + cy = x^3 + ax + b \qquad a, b, c \in K.$$

Loi de groupe

Soient P_1, P_2 deux points d'une courbe elliptique E et D la droite passant par ces deux points (et tangente à E si $P_1 = P_2$). Par le théorème de Bézout, il existe un unique autre point P_3 (en tenant compte des multiplicités éventuelles) d'intersection de D avec E (à priori, les coordonnées de P_3 sont dans une extension de K, mais le calcul de ses coordonnées en fonction de celles de P_1 et P_2 montre qu'elles sont aussi dans K).

Lorsque aucun de ces trois points n'est à l'infini, on peut noter (x_i, y_i) les coordonnées affines des P_i et le calcul montre que les coordonnées de P_3 sont données par

$$\begin{cases} x_3 = m^2 - x_1 - x_2, \\ y_3 = y_1 + m(x_3 - x_1) \end{cases} \qquad \text{où} \quad m = \begin{cases} \dfrac{y_1 - y_2}{x_1 - x_2} & \text{si } x_1 \neq x_2 \\ \dfrac{3x_1^2 + a}{2y_1} & \text{si } P_1 = P_2 \end{cases} \tag{11}$$

Notons O le point distingué de E. On montre qu'il existe une loi de groupe abélien sur les points de E telle que O soit l'élément neutre et que l'identité $P_1 + P_2 + P_3 = O$ soit vérifiée pour tout triplet de points alignés. Lorsque la convention $O = (0:1:0)$ est en vigueur, l'opposé de chaque point de E est son symétrique par rapport à l'axe des abscisses. La somme de deux points P_1, P_2 finis et non opposés est donc le point de coordonnées $(x_3, -y_3)$ où x_3, y_3 sont donnés par (11). Bien sûr, lorsque l'un des points est à l'infini, ou lorsque ils sont symétriques l'un de l'autre, on a les identités $P + O = P$ et $P + (-P) = O$.

6.2. — Exercice. Soient $(G, +)$ un groupe abélien, d'élément neutre 0, et e un élément quelconque de G. On pose $a \oplus b = a + b - e$. Montrer que $\oplus$ est aussi une loi de groupe abélien sur G et que son élément neutre est e.

Isogénies

6.3. — Définition. Soient E et E' deux courbes elliptiques sur un corps K. Une *isogénie* (homomorphisme de courbes elliptiques) de E dans E' est une application rationnelle ϕ (homomorphisme de courbes algébriques) non identiquement nulle et telle que $\phi(O) = O'$. On démontre alors que ϕ est un homomorphisme de groupes.

6.4. — Exemple. Soient E une courbe elliptique et m entier. la multiplication par m (au sens de la loi de groupe) est une isogénie de E dans E, souvent notée $[m]$.

6.5. — Proposition et définition. Soit $\phi : E \longrightarrow E'$ une isogénie. Alors ϕ est un homomorphisme de groupes et son noyau est un sous-groupe fini de E. L'ordre de $\ker \phi$ est appelé le degré de ϕ.

6.6. — Proposition et définition. Soit $\phi : E \longrightarrow E'$ une isogénie de degré d. Alors il existe une unique isogénie $\hat{\phi} : E' \longmapsto E$ telle que $\hat{\phi} \circ \phi = [d]$. On l'appelle *l'isogénie duale de* ϕ.

Isomorphismes

Un isomorphisme entre deux courbes elliptiques E et E' est une isogénie bijective. Si on considère seulement des courbes elliptiques sous "forme courte" de Weierstrass :

$$y^2 = x^3 + ax + b \qquad \text{et} \qquad y'^2 = x'^3 + a'x' + b',$$

alors tout isomorphisme est de la forme

$$x' = u^2 x \quad \text{et} \quad y' = u^3 y \qquad \text{avec } u \in K^*$$

et on a les relations

$$u^4 a' = a \quad \text{et} \quad u^6 b' = b.$$

Courbes elliptiques sur $\mathbb{C}$ et réseaux

6.7. — Définition. On appelle *réseau* de $\mathbb{C}$ un sous-groupe de $\mathbb{C}$ discret et de rang 2. Autrement dit, c'est une partie de la forme

$$\{m\omega_1 + n\omega_2 \mid m, n \in \mathbb{Z}\} \qquad \text{où } \omega_1, \omega_2 \in \mathbb{C} \text{ sont } \mathbb{R}\text{-linéairement indépendants.}$$

6.8. — Définition. Une *fonction elliptique* associée au réseau L est une fonction méromorphe sur $\mathbb{C}$ L-périodique, i.e. telle que $f(z+\omega) = f(z)$ pour tous $z \in \mathbb{C}, \omega \in L$.

6.9. — Définition. La *fonction de Weierstrass* associée au réseau L est la somme suivante

$$\wp(z) = \wp(z, L) = \frac{1}{z^2} + \sum_{\omega \in L - \{0\}} \left[\frac{1}{(z-\omega)^2} - \frac{1}{\omega^2} \right].$$

Elle converge uniformément sur tout compact de $\mathbb{C} \setminus L$. Elle est méromorphe sur $\mathbb{C}$. Ses pôles sont aux points de L, doubles et de résidu 0. Sa dérivée est donnée par

$$\wp'(z) = \wp'(z, L) = -2 \sum_{\omega \in L} \frac{1}{(z-\omega)^3}.$$

En particulier :

6.10. — Proposition. *les fonctions $\wp$ et $\wp'$ sont des fonctions elliptiques associées au réseau L.*

6.11. — Proposition. *La fonction $\wp$ vérifie l'équation différentielle*

$$\wp'(z)^2 = 4\wp(z)^3 - 60G_2\wp(z) - 140G_3 \qquad \text{où} \quad G_k = G_k(L) = \sum_{\omega \in L - \{0\}} \frac{1}{\omega^{2k}}.$$

Ainsi, à tout réseau L de $\mathbb{C}$, est associée une courbe elliptique E_L d'équation $y^2 = 4x^3 - 60G_2x - 140G_3$. Réciproquement, le résultat suivant montre que chaque courbe elliptique sur $\mathbb{C}$ est associée à un unique réseau.

6.12. — Théorème (d'uniformisation). *Soient $g_2, g_3 \in \mathbb{C}$ tels que $g_2^3 - 27g_3^2 \neq 0$. Il existe un unique réseau L de $\mathbb{C}$ tel que*

$$g_2 = 60G_2(L) \qquad \text{et} \qquad g_3 = 140G_3(L).$$

6.13. — Proposition. Soient E une courbe elliptique sur $\mathbb{C}$ et L le réseau correspondant. L'application

$$\mathbb{C}/L \ni z \longmapsto (\wp(z), \wp'(z)) \in E$$

est un isomorphisme de groupes analytiques.

Courbes elliptiques sur un corps fini

6.14. — Exercice. Soit E une courbe elliptique d'équation (10) sur F_p avec p premier. Le nombre de points de E est donné par

$$\#E = 1 + \sum_{x \in \mathbb{F}_p} \left(1 + \left(\frac{x^3 + ax + b}{p} \right) \right).$$

Si le corps de base est un corps fini, la courbe elliptique est un groupe fini et le théorème suivant donne un renseignement très utile sur son ordre.

6.15. — Théorème (Hasse). *Soit E une courbe elliptique sur le corps fini $\mathbb{F}_q$. Le nombre de points de E vérifie*

$$|\#E - (q+1)| \leqslant 2\sqrt{q}.$$

D'autre part, deux éléments suffisent pour engendrer le groupe d'une courbe elliptique et, bien souvent, il est même cyclique. Précisément, on a

6.16. — Théorème. *Soit E une courbe elliptique sur le corps fini $\mathbb{F}_q$. Alors, le groupe E est cyclique ou bien isomorphe à $\mathbb{Z}/n_1\mathbb{Z} \times \mathbb{Z}/n_2\mathbb{Z}$ avec $n_2 \mid \operatorname{pgcd}(n_1, q-1)$.*

Couplages

Soit E une courbe elliptique sur un corps fini $\mathbb{F}_q$. Pour $\ell \geqslant 2$, on note $E[\ell]$ le sous-groupe de ℓ-torsion de E, c'est-à-dire :

$$\{P \in E \mid \ell P = 0\}.$$

6.17. — Définitions. Un *couplage d'ordre ℓ sur E* est une application

$$\langle \cdot, \cdot \rangle : \begin{cases} E[\ell] \times E[\ell] \longrightarrow \mathbb{U}_\ell \subseteq \mathbb{F}_q^* \\ (P, Q) \longmapsto \langle P, Q \rangle. \end{cases}$$

où $\mathbb{U}_\ell$ est le sous-groupe formé des racines ℓ-ièmes de l'unité, dans une extension convenable de $\mathbb{F}_q$ (en fait, dans $\mathbb{F}_{q^r}$ où r est l'ordre de q modulo ℓ). Un tel couplage est dit *bilinéaire* si on a les identités

$$\langle P+Q, R\rangle = \langle P, R\rangle\langle Q, R\rangle \quad \text{et} \quad \langle P, Q+R\rangle = \langle P, Q\rangle\langle P, R\rangle \qquad \text{pour tous } P, Q, R \in E[\ell].$$

Il est dit *symétrique* si on a l'identité $\langle P, Q\rangle = \langle Q, P\rangle$ pour tous $P, Q \in E[\ell]$, et il est dit *non-dégénéré* si, pour tout $P \in E[\ell]$ non nul, il existe un $Q \in E[\ell]$ tel que $\langle P, Q\rangle \neq 1$.

6.18. — Proposition. *Soit $\langle \cdot, \cdot \rangle$ un couplage bilinéaire d'ordre l sur une courbe elliptique E. On a les identités suivantes.*

- $\langle P, 0\rangle = 1 = \langle 0, P\rangle$.
- $\langle -P, Q\rangle = \langle P, Q\rangle^{-1} = \langle P, -Q\rangle$.

Démonstration — Exercice. □

En cryptographie, deux (au moins) couplages bilinéaires ont des applications : le couplage de Weil et celui de Tate. Le couplage de Weil est symétrique et a pour propriété que $\langle P, P\rangle = 1$ pour tout $P \in E[\ell]$.

Chapitre 7

Notions de Théorie de la Complexité

Le but est de classifier les problèmes algorithmiques, en fonction de leur difficulté (i.e. du nombre d'opérations élémentaires nécessaires pour les résoudre). L'une des premières étapes est de donner un sens précis à "opération élémentaire". Pour cela, il faut définir un modèle d'ordinateur. Le modèle le plus utilisé est celui de la machine de Turing, qui est un modèle très simple mais donnant une approximation raisonnable (du point de vue de la théorie de la complexité) d'un ordinateur réel. Comme références sur ce chapitre, citons les livres de Aho, Hopcroft & Hullman [10], de Manna [75], les articles de Boas [22], de Cook [35] et bien sûr, celui de Turing [128].

Notons que la théorie de la complexité est un domaine vaste et plein de subtilités. Son utilité est majeure pour qui veut comprendre certaines facettes de la cryptographie et de l'algorithmique mais la compréhension de ce chapitre n'est pas indispensable à la lecture des suivants...

1. Machines de Turing

1.1. — Description. Une *machine de Turing* est la donnée de

• Un alphabet Σ (i.e. un ensemble fini dont les éléments sont appelés *symboles*). On posera $\Sigma' = \Sigma \cup \{\not b\}$ où $\not b$ est un symbole supplémentaire, le "caractère blanc".

• Une bande infinie (dans les deux sens) formée de cases juxtaposées linéairement, destinées à être marquées par des symboles. Cette bande symbolise la mémoire d'un ordinateur. Les symboles écrits dessus (un symbole par case) représenteront les données du programme et leurs évolutions au cours du calcul. La bande est munie d'une tête de lecture/écriture qui se déplacera de case en case adjacente.

• Un ensemble fini Q d'états, l'un d'entre eux étant appelé l'état *initial*.

• Un programme ou *fonction de transition*. Ce programme est composé d'un tableau d'instructions (ou transitions), indexé par Q et Σ'. Pour chaque couple $(q, s) \in Q \times \Sigma'$, le programme possède au plus une instruction qui sera exécutée quand la machine sera dans l'état q et que la tête lira le symbole s. Cette instruction est codée sous la forme (q', s', d) avec $q' \in Q$, $s' \in \Sigma'$ et $d \in \{droite, gauche\}$. L'exécution de cette instruction consiste à écrire le symbole s' (à la place du symbole s), à déplacer la tête d'une case dans la direction (droite ou gauche) indiquée, et à placer la machine dans l'état q'.

Calcul par une machine de Turing

Pour faire fonctionner la machine de Turing, on inscrit des *données* sur la bande, sous la forme de mots de Σ^*. La taille totale des données est finie. Leurs différents éléments sont séparés par des blancs $\not b$. Les cases non-utilisées par les données sont aussi remplies par des blancs. On place la tête de lecture sur le premier symbole de la donnée (extrémité gauche) et on met la machine dans l'état initial. Ensuite, la machine exécute le programme et s'arrête lorsque elle ne possède pas d'instruction correspondant au symbole lu et à l'état où elle se trouve. Le *résultat* du programme est constitué des mots alors inscrits sur la bande.

1.2. — Exemple (incrémenteur). Cette machine additionne 1 à la donnée présentée en entrée, écrite en binaire.

$$\Sigma = \{0, 1\}, \qquad Q = \{q_0, q_1, q_2\}, \quad \text{l'état initial étant } q_0.$$

Sa fonction de transition est donnée par le tableau suivant :

	0	1	$\not b$
q_0	$(q_0, 0, droite)$	$(q_0, 1, droite)$	$(q_1, \not b, gauche)$
q_1	$(q_2, 1, gauche)$	$(q_1, 0, gauche)$	$(q_2, 1, gauche)$

Certaines machines de Turing sont conçues pour que cette situation d'arrêt survienne exactement lorsque celles-ci atteignent un état dit *final*. Plus précisément, ces machines possèdent un ou plusieurs états finaux à partir desquels aucune transition n'est possible, et les états non-finaux possèdent des transitions pour chaque symbole de Σ'. (c'est le cas de l'exemple 1.2, qui possède un unique état final q_2).

1.3. — Exercice. Montrer que n'importe quelle machine de Turing est équivalente (i.e. génére les mêmes résultats à partir des même données) à une machine de Turing à un état final.

Comme un ordinateur réel, une machine de Turing contient à chaque instant une quantité finie d'information en mémoire et n'utilise, lors de son fonctionnement, qu'une longueur finie de sa bande (si le programme se termine). Contrairement à un ordinateur réel, la capacité de la mémoire d'une machine de Turing est infinie. Cela peut paraître un défaut du modèle, mais dans la réalité, lorsqu'un ordinateur est trop petit pour résoudre un problème, n'a-t-on pas la possibilité de le remplacer par un ordinateur plus gros ?

Machines de Turing à plusieurs états finaux

Dans certaines applications, ce n'est pas au résultat écrit sur la bande après l'arrêt de la machine auquel on s'intéresse, mais plutôt à l'état dans laquelle la machine s'arrête. Par exemple, la machine suivante possède deux états finaux ayant chacun une signification. Par contre, elle ne calcule pas de résultat (si on s'en tient à la terminologie introduite ci-dessus) car elle ne modifie pas le contenu de la bande.

1.4. — Exemple (testeur de poids pair). Cette machine s'arrête dans un état différent selon que le nombre binaire entré possède un nombre pair (état q_y) ou impair (état q_n) de 1.

$$\Sigma = \{0,1\}, \qquad Q = \{q_0, q_1, q_y, q_n\}, \quad \text{l'état initial étant } q_0.$$

Sa fonction de transition est donnée par le tableau suivant :

	0	1	$\not\emptyset$
q_0	$(q_0, 0, droite)$	$(q_1, 1, droite)$	$(q_y, \not\emptyset, droite)$
q_1	$(q_1, 0, droite)$	$(q_0, 1, droite)$	$(q_n, \not\emptyset, droite)$

L'importance théorique des machines de Turing

Quels sont les problèmes que peuvent traiter les machines de Turing ? Tous les problèmes qui peuvent être résolus à l'aide d'un ordinateur réel, actuel ou futur (avec peut-être pour exception certains ordinateurs "quantiques" si jamais ceux-ci deviennent un jour réalité), programmé dans n'importe quel langage, peuvent en principe être résolus par une machine de Turing (munie d'un générateur aléatoire). Bien que ce fait ne puisse être prouvé rigoureusement (le concept d'ordinateur réel est trop informel, et l'intérêt principal des machines de Turing est justement de les modéliser) de nombreux travaux ont étayé cette hypothèse. On particulier, de nombreuses autres tentatives de modélisation d'ordinateurs (lambda calcul, fonctions récursives, etc) ont vu le jour, et elles se sont ensuite révélées être équivalentes au modèle de Turing.

De plus, on peut imaginer des dispositifs apparemment plus puissants que les machines de Turing, en autorisant plusieurs bandes et/ou plusieurs têtes de lecture/écriture notamment. Mais l'examen montre que les problème que peuvent résoudre ces dispositifs peuvent aussi l'être par des machines de Turing classiques. Inversement, on montre que toute machine de Turing peut être simulée par une machine de Turing dont l'alphabet est réduit à $\{0,1\}$ (voire à $\{1\}$ au prix d'une dégradation majeure en complexité). On montre aussi qu'on ne perd pas de potentialités si on impose que la bande soit finie dans l'un des deux sens. Bref la machine de Turing, malgré son apparence simpliste et sa difficulté à programmer, s'est largement imposée comme modèle théorique pour analyser les algorithmes et leur complexité.

Machine de Turing universelle

On peut remarquer qu'une machine de Turing peut être décrite à l'aide d'un nombre fini de symboles. On adoptant un encodage convenable, on peut même représenter chaque machine de Turing par un mot de $\{0,1\}^*$. Rien n'empêche alors d'utiliser la représentation d'une machine de Turing comme donnée d'une autre machine de Turing sur ce même alphabet. En particulier, il existe une machine de Turing dite *universelle* qui, lorsque on lui fournit comme données la description d'une machine M et d'autres données D, simule l'exécution de M sur les données D, le résultat obtenu étant identique à celui qu'on aurait en appliquant la machine M aux données D.

2. Extensions du concept de machine de Turing

Machines de Turing non déterministes

C'est une **extension** du concept de machine de Turing, qui sera utile pour étudier les algorithmes non-déterministes ainsi que les problème **NP**. Une *machine de Turing non déterministe*, c'est comme une machine de Turing sauf que plusieurs transitions peuvent correspondre à certains couples de $Q \times \Sigma'$. Sans réduire la puissance du concept, on peut supposer qu'au plus deux transitions peuvent correspondre au même couple.

On considère que lorsque une machine de Turing non déterministe exécute un calcul et qu'elle à le choix entre plusieurs transitions possibles, elle se duplique pour que chaque séquence de choix possible soit prise par l'un des exemplaires de la machine. On s'intéresse alors au calcul effectué (ou plutôt à l'état final atteint) par le premier exemplaire qui s'arrête (s'il y en a au moins un qui s'arrête). Cela permettra de définir les problèmes **NP**. Clairement, une machine de Turing non-déterministe est un concept formel qui rompt avec le rôle de modèle d'ordinateur réel.

Machines de Turing probabilistes

C'est un **autre extension** du concept de machine de Turing. La différence avec l'extension précédente est subtile mais importante.

La description d'une machine de Turing probabiliste est identique à celle d'une machine non déterministe, c'est-à-dire une machine pour laquelle deux transitions peuvent correspondre au même couple de $Q \times \Sigma'$.

La différence intervient dans la façon dont elle est censée exécuter son programme. Lorsque une machine de Turing probabiliste a le choix entre deux transitions possibles au cours d'un calcul, elle choisit l'une d'entre elle au hasard et poursuit son calcul par cette transition. Le fonctionnement de la machine (en particulier le résultat et le temps de calcul) peut être différent selon la séquence de choix aléatoires choisie, pour une même donnée.

La machine de Turing probabiliste n'est pas un modèle d'ordinateur dans le sens où ce dernier est un appareil qui se veut parfaitement déterministe. Toutefois, le modèle de machine de Turing probabiliste est essentiel pour analyser toute une classe d'algorithmes (les algorithmes *probabilistes*) dont l'utilité est bien réelle et qui utilisent des tirages aléatoires lors de leur fonctionnement. Dans la pratique, ces tirages aléatoires sont simulés dans un ordinateur réel par un générateur pseudo-aléatoire qui est un dispositif déterministe mais dont le comportement "ressemble" à de l'aléa.

3. Décidabilité

Langage reconnu par une machine de Turing

Soit Σ un alphabet. On note Σ^* l'ensemble des mots sur Σ. Un *langage sur* Σ est une partie de Σ^*.

3.1. — Définitions. Soit M une machine de Turing sur l'alphabet Σ et q_y un état de M. Pour $x \in \Sigma^*$, on dit que la machine de Turing M *accepte la donnée* x (par l'état q_y) si M s'arrête dans l'état q_y lorsque sa donnée est x. L'ensemble de mots acceptés par M s'appelle le *langage reconnu par* M. On le notera $L_{q_y}(M)$ ou $L(M)$.

Le complémentaire de $L(M)$ dans Σ^* contient donc les mots pour lesquels la machine M s'arrête dans un état différent de q_y ainsi que ceux pour lesquels M ne s'arrête pas.

3.2. — **Exemple.** La machine de Turing suivante reconnaît le langage des mots binaires ayant au moins un 1 :

	0	1	$\not\emptyset$
q_0	$(q_0, 0, droite)$	$(q_f, 1, droite)$	$(q_1, \not\emptyset, gauche)$
q_1	$(q_1, 0, gauche)$	$(q_f, 1, gauche)$	$(q_0, \not\emptyset, droite)$

3.3. — **Définitions.** Soit M une machine de Turing non déterministe sur l'alphabet Σ et q_y un état de M. Pour $x \in \Sigma^*$, on dit que la machine M *accepte la donnée* x (par l'état q_y) si M s'arrête dans l'état q_y lorsque sa donnée est x, pour au moins une séquence de choix. Comme dans le cas déterministe, l'ensemble de mots acceptés par M s'appelle le langage *reconnu par* M et on le notera $L_{q_y}(M)$ ou $L(M)$.

Le complémentaire de $L(M)$ dans Σ^* contient donc les mots pour lesquels, quelle que soit la séquence de choix, la machine M s'arrête dans un état différent de q_y ou bien ne s'arrête pas.

Problèmes de décision

Par simplicité, on se restreint souvent en théorie de la complexité aux *problèmes de décision*, c'est-à-dire aux problèmes dont la réponse (pour chaque donnée) ne peut prendre que deux valeurs, en général *oui* ou *non*.

Pour qu'une machine de Turing puisse résoudre un tel problème, il faut choisir un alphabet Σ ainsi qu'une représentation des données du problème, c'est-à-dire une application injective π de l'ensemble des données possibles dans Σ^*. L'image D de π est un langage sur Σ (non nécessairement égal à Σ^*) et l'ensemble D^+ des $\pi(x)$ pour lesquels x appelle la réponse *oui* est un langage contenu dans D. De même pour l'ensemble $D^- = D \setminus D^+$ des $\pi(x)$ pour lesquels x appelle la réponse *non*. En identifiant les données x du problème à leurs images $\pi(x)$ dans Σ^*, tout problème de décision revient à discerner dans un langage D ceux qui sont dans un sous-langage D^+ de D.

3.4. — **Définition.** Soit $\mathcal{D}$ un problème de décision, D le langage formé de l'ensemble des représentations des données de $\mathcal{D}$ et D^+ le sous-ensemble de D correspondant aux données appelant la réponse *oui*. On dit qu'une machine de Turing M, déterministe ou non déterministe, *résout* $\mathcal{D}$ si elle possède un état q_y pour lequel $L_{q_y}(M) \cap D = D^+$.

Une machine de Turing (déterministe) qui résout $\mathcal{D}$ est une machine qui s'arrête dans l'état q_y si et seulement si la donnée appartient à D^+. Une machine de Turing non déterministe qui résout $\mathcal{D}$ est une machine qui s'arrête dans l'état q_y pour au moins une séquence de choix si et seulement si la donnée appartient à D^+. On notera que ces définitions n'imposent rien sur le comportement de ces machines de Turing lorsque la donnée appartient à D^-. Elle peut alors s'arrêter dans un état différent de q_y mais elle peut tout aussi bien ne pas s'arrêter du tout. De plus, même lorsque la donnée appartient à D^+, dans le cas non déterministe, cette machine peut ne pas s'arrêter, voire s'arrêter dans un état autre que q_y, pour certaines séquences de choix.

3.5. — **Exemple.** On considère le problème de décision dont les données x possibles sont les éléments de $\mathbb{N}^*$ et qui pose la question "x est-t-il multiple de 3 ?". Les données peuvent être représentées par leur développement binaire, qui est un mot sur $\{0, 1\}$, et D^+ est l'ensemble des développements binaires correspondant à des multiples de 3.

3.6. — **Exercice.** Construire une machine de Turing résolvant le problème de décision précédent.

Problèmes décidables

3.7. — **Définition.** Soient $\mathcal{D}$ un problème de décision dont les données sont représentées par un langage D. On note $\overline{\mathcal{D}}$ le *problème de décision complémentaire à* $\mathcal{D}$ tel que $\overline{D}^+ = D^-$ (notations évidentes).

3.8. — **Définitions.** On dit qu'un problème de décision $\mathcal{D}$ est *partiellement décidable* s'il existe une machine de Turing (déterministe) qui résout $\mathcal{D}$. On dit que $\mathcal{D}$ est *décidable* si $\mathcal{D}$ et $\overline{\mathcal{D}}$ sont partiellement décidables. On dit que $\mathcal{D}$ est *indécidable* s'il n'est pas décidable (même s'il est partiellement décidable).

3.9. — Exercice. Montrer qu'un problème de décision $\mathcal{D}$ est décidable si et seulement si il existe une machine de Turing M munie de deux états q_y et q_n telle que $L_{q_y}(M) = D^+$ et $L_{q_n}(M) = D^-$.

3.10. — Exemple (problème de l'arrêt des machines de Turing). On considère le problème de décision suivant. Soit $\Sigma = \{0, 1\}$. Etant donnée une machine de Turing M sur Σ et $w \in \Sigma^*$, la machine M s'arrête-t-elle lorsque sa donnée est w ?

3.11. — Théorème (indécidabilité). Le problème de l'arrêt des machines de Turing est indécidable.

Démonstration — Supposons que ce problème soit décidable. Il existerait alors une machine de Turing A qui, à partir de la description d'une machine M et d'une donnée d s'arrêterait après un temps fini et indiquerait si la machine M munie de la donnée d s'arrête. On pourrait alors construire une machine B exécutant le programme suivant lorsqu'on lui donne en entrée la description d'une machine de Turing M :

$$B(M) : \qquad \text{Si } A(M, M) = \text{oui, alors entrer dans une boucle infinie, sinon stop.}$$

En appliquant la machine B à sa propre description, on obtient une contradition. Par construction de B, l'instance $B(B)$ se termine si et seulement si la réponse à $A(B, B)$ est "non". Mais par définition de A, cette même instance se termine si et seulement si la réponse à $A(B, B)$ est "oui". □

3.12. — Exercice. Le problème de l'arrêt des machines de Turing est partiellement décidable. (Appliquer la machine universelle.)

3.13. — Complément. Ce problème admet une variante uniforme : Etant donnée une machine de Turing sur Σ, s'arrête-t-elle toujours quelle que soit l'entrée $w \in \Sigma^*$? Cette variante n'est pas partiellement décidable ni son problème conjugué.

3.14. — Exemple (dixième problème de Hilbert). En 1901, Hilbert a énuméré une liste de problèmes alors non résolus et qu'il considérait comme particulièrement importants. Le dixième d'entre eux était de trouver un algorithme qui prenne en entrée un polynôme $P(X_1, \ldots, X_k)$ à coefficients entiers et qui détermine si l'équation $P(x_1, \ldots, x_k) = 0$ admet une solution en nombres entiers. Matijasevič a montré en 1970 qu'un tel algorithme n'existe pas. Le dixième problème de Hilbert est donc indécidable.

4. Complexité des algorithmes déterministes

Coûts

Si on veut exécuter un algorithme sur une donnée d'entrée x, deux coûts sont à considérer :

- Le coût en temps $\mathcal{C}_{\text{temps}}(x)$. C'est le nombre d'opérations effectuées pour obtenir le résultat final.
- Le coût en espace $\mathcal{C}_{\text{espace}}(x)$. C'est la taille de la mémoire nécessaire pour mener à bien le calcul.

En modélisant les ordinateurs par les machines de Turing, on donne un sens précis à ces coûts. Le coût en temps est le nombre de déplacements de la tête de lecture avant l'arrêt de la machine. Le coût en espace est le nombre de cases écrites au moins une fois. On notera que le coût en temps est toujours supérieur ou égal au coût en espace.

Les coûts d'un algorithme sont des fonctions de la taille n de la donnée d'entrée x. Sont intéressants les coûts maximaux :

$$\left.\begin{aligned} \mathcal{C}^{\max}_{\text{temps}}(n) &= \max_x \mathcal{C}_{\text{temps}}(x) \\ \mathcal{C}^{\max}_{\text{espace}}(n) &= \max_x \mathcal{C}_{\text{espace}}(x) \end{aligned}\right\} \quad x \text{ parcourant l'ensemble des données de longueur } \leqslant n,$$

et les coûts moyens :

$$\left.\begin{aligned} \mathcal{C}^{\text{moy}}_{\text{temps}}(n) &= \operatorname*{moy}_x \mathcal{C}_{\text{temps}}(x) \\ \mathcal{C}^{\text{moy}}_{\text{espace}}(n) &= \operatorname*{moy}_x \mathcal{C}_{\text{espace}}(x) \end{aligned}\right\} \quad x \text{ parcourant l'ensemble des données de longueur } n,$$

la longueur d'une donnée étant bien sûr le nombre de cases qu'elle occupe sur la bande.

Taux de croissance

Pour comparer les taux de croissance de fonctions, nous emploierons les notations suivantes :

4.1. — Notations. Soient f, g deux fonctions $\mathbb{N} \to \mathbb{R}$, croissantes et positives.
- $f = O(g)$ s'il existe $c \in \mathbb{R}$ tel que $f(n) \leqslant cg(n)$ pour n assez grand.
- $f = \Omega(g)$ s'il existe $c \in \mathbb{R}_+^*$ tel que $f(n) \geqslant cg(n)$ pour n assez grand.
- $f = \Theta(g)$ si $f = O(g)$ et $f = \Omega(g)$.
- $f = o(g)$ s'il existe une fonction $\varepsilon : \mathbb{N} \mapsto \mathbb{R}$ tendant vers 0 et telle que $f = \varepsilon g$.
- $f \sim g$ si la limite de f/g vaut 1.

4.2. — Exercice. Montrer les relations suivantes.
- Si f est une fonction polynomiale de degré k et de coefficient dominant positif alors $f = \Theta(n^k)$.
-
$$n! = o(n^n), \qquad n! = \Omega(2^n), \qquad \log(n!) = \Theta(n \log(n)).$$

Complexité polynomiale / non polynomiale

Quel coût est acceptable pour un problème donné ? Il n'y a jamais de réponse claire car tout dépend des moyens dont on dispose et du temps qu'on est prêt à attendre. Tout juste peut-on dire de façon très vague qu'un coût de l'ordre de 2^{40} opérations "élémentaires" est accessible à un particulier s'il est un peu patient, que 2^{56} opérations devient accessible si on dispose de moyens à grande échelle (e.g. cryptanalyse de DES) et qu'un coût de 2^{80} opérations "très élémentaires" est hors de portée de quiconque.

Le comportement du coût d'un algorithme quand la taille de la donnée tend vers l'infini est presque toujours un très bon indicateur du comportement de l'algorithme sur des données de taille utile. Par exemple, pour déplacer une tour de Hanoï de N disques, il faut $2^N - 1$ déplacements de disques ce qui est un coût exponentiel. Pour $N = 10$ le nombre de déplacements de disques est déjà 1023, et pour $N = 20$ il est 1048575...Le déplacement d'une tour est d'un coût prohibitif pour des valeurs de N très faibles.

Soit un algorithme dont le nombre d'opérations en fonction de la taille n de l'entrée est décrit par la fonction f. Si on dispose d'un ordinateur qui fait 10^9 opérations par secondes, voici le temps pris par l'algorithme pour quelques instances de f et de n.

f	$\log_2(n)$	n	$n\log_2(n)$	$100n$	n^2	n^3	n^6	2^n
$n = 100$	$6.6 \cdot 10^{-9}$ s	10^{-7} s	$6.6 \cdot 10^{-7}$ s	10^{-5} s	10^{-5} s	10^{-3} s	17 min	$4 \cdot 10^{13}$ ans
$n = 10^5$	$1.7 \cdot 10^{-8}$ s	10^{-4} s	$1.7 \cdot 10^{-3}$ s	0.01 s	10 s	11.5 jours	$\geqslant 10^{13}$ ans	$\geqslant 10^{30086}$ans
$n = 10^{10}$	$3.3 \cdot 10^{-8}$ s	10 s	332 s	17 min	3 siècles	$\geqslant 10^{13}$ans	$\geqslant 10^{46}$ ans	$\geqslant 10^{3 \cdot 10^9}$ans
$n = 10^{20}$	$6.6 \cdot 10^{-8}$ s	30 siècles	$\geqslant 10^5$ ans	$\geqslant 10^5$ans	$\geqslant 10^{23}$ans	$\geqslant 10^{43}$ans	$\geqslant 10^{103}$ans	!!!

Un algorithme est considéré comme efficace si sa complexité en temps est polynomiale (c'est-à-dire majorée par une fonction polynôme).

Classes **P** et **NP**

4.3. — Définitions. • Un problème de décision appartient à *la classe* **P** s'il existe une machine de Turing (déterministe) qui le résout en temps polynomial (lorsque la donnée appartient à D^+, la machine s'arrête dans l'état q_y en un temps majoré par une fonction polynomiale de la taille de la donnée).

• Un problème de décision appartient à *la classe* **NP** s'il existe une machine de Turing non déterministe qui le résout en temps polynomial (lorsque la donnée appartient à D^+, la machine s'arrête dans l'état q_y et en un temps majoré par une fonction polynomiale de la taille de la donnée, pour au moins une séquence de choix.

• Un problème de décision $\mathcal{D}$ appartient à *la classe co-***P** ou à *la classe co-***NP** si le problème de décision complémentaire $\overline{\mathcal{D}}$ appartient respectivement à la classe **P** ou à la classe **NP**.

4.4. — Théorème. *Les classes* **P** *et co-***P** *sont égales.*

Démonstration — Il suffit de montrer que $\mathbf{P} \subseteq$ co-**P**. Soit $\mathcal{D}$ un problème de décision dans la classe **P**. Il existe donc une machine de Turing M_1 qui résout $\mathcal{D}$ en temps polynomial $P(m)$. Il est alors possible de construire une machine de Turing M_2 qui simule le fonctionnement de M_1 mais qui en plus gère un compteur qui compte les pas de M_1. Cette machine M_2 peut être construite de telle manière que le coût de la gestion du compteur soit majoré par $mP(m)$ pour m assez grand. Lorsque la valeur du compteur dépasse $P(m)$, la machine M_2 passe dans un état spécifique et s'arrête. La machine M_2 résout le problème $\overline{\mathcal{D}}$ en temps polynomial. Elle montre donc que $\mathcal{D} \in$ co-**P**. □

4.5. — Proposition. *La classe* **P** *est contenue dans les classes* **NP** *et co-***NP**.

Démonstration — C'est clair. □

Les conjectures suivantes sont encore non démontrées

$$\mathbf{P} \neq \mathbf{NP}, \qquad \mathbf{NP} \neq \text{co-}\mathbf{NP}, \qquad \mathbf{P} \neq \mathbf{NP} \cap \text{co-}\mathbf{NP}.$$

Réduction polynomiale

4.6. — Définition. Soient L_1 et L_2 deux langages sur un alphabet Σ. On dit que L_1 *est polynomialement réductible à* L_2 s'il existe une machine de Turing d'alphabet Σ qui, lorsqu'on la fait fonctionner sur une donnée quelconque $\in \Sigma^*$, s'arrête au bout d'un temps polynomial (i.e. majoré par une fonction polynomiale de la longueur de l'entrée) en laissant sur la bande un élément de L_2 si et seulement si la donnée appartenait à L_1.

4.7. — Définition. Soient $\mathcal{D}_1$ et $\mathcal{D}_2$ deux problèmes de décision. On dit que $\mathcal{D}_1$ *est polynomialement réductible à* $\mathcal{D}_2$ s'il existe un algorithme A_1 utilisant éventuellement comme sous-algorithme une procédure A_2 résolvant $\mathcal{D}_2$ et vérifiant les condition suivantes.

• Le nombre d'appels de A_2 par A_1 est majoré par une fonction polynomiale de la donnée.

• Le coût de A_1, hors coût interne de A_2, est polynomial.

Cela revient à dire, que si on considère le coût de A_2 comme polynomial, alors le coût de A_1 est aussi polynomial. On pourrait craindre que A_1, utilisant une quantité polynomiale d'appels à A_2 mais avec des paramètres arbitrairement grands ne soit pas polynomial. Mais si le deuxième point est vérifié alors la taille maximale des paramètres transmis à A_2 par A_1 est polynomiale, donc A_1 est polynomial.

4.8. — Définition. On dit que deux problèmes de décision sont *polynomialement équivalents* s'ils sont mutuellement polynomialement réductibles l'un à l'autre.

Problèmes **NP**-Complets

4.9. — Définitions. Un problème de décision $\mathcal{D}$ est dit **NP**-*Dur* si tout problème de décision de la classe **NP** est polynomialement réductible à $\mathcal{D}$. Un problème de décision **NP**-*Complet* est un problème de classe **NP** et **NP**-Dur.

4.10. — Exemple. Problème de satisfiabilité (SAT). Soient $x_1, \ldots, x_n$ des variables booléennes. Une clause booléenne est une expression de la forme (y_1 ou y_2 ou ... ou y_k) où chaque y_k est une variable x_i ou bien $\overline{x}_i$. Le problème SAT est le problème de décision suivant. Etant donnée une liste finie de clauses, existe-t-il des valeurs booléennes qui, substituées aux variables, font prendre simultanément à chaque clause de la liste la valeur *vrai* ?

Si on affecte des valeurs booléennes à chaque variable, il est très facile de vérifier si les clauses sont satisfaites. Le problème SAT est donc de classe **NP** (la taille d'une instance du problème SAT est la somme de la taille de ses clauses et la taille d'une clause est le nombre de variables qu'elle contient).

4.11. — Théorème (Cook). *Le problème SAT est* **NP***-Complet.*

Démonstration — Voir [35]. Dans les grandes lignes, la démonstration est la suivante. Il s'agit de montrer que tout problème de décision $\mathcal{D}$ de classe **NP** est polynomialement réductible au problème SAT. Soit M une machine de Turing non-déterministe résolvant $\mathcal{D}$. Pour chaque donnée appelant la réponse "oui", il existe une séquence de transitions de la machine M menant de la configuration initiale de M à une configuration dans l'état d'acceptation. Chaque configuration de la machine M (contenu de la bande, état courant, position de la tête de lecture) peut être codée par une suite de symboles dans un alphabet fini (la réunion de l'alphabet de M et de l'ensemble de ses états). On peut ainsi coder chaque configuration que prend la machine M au cours du "calcul" menant à l'acceptation d'une donnée, et mettre les codes obtenus les uns au dessous des autres dans l'ordre chronologique. On obtient alors un tableau de symboles, de taille finie. Considérons les variables booléennes $X_{i,j,s}$ indexées par les lignes et colonnes (i,j) du tableau et par l'ensemble des symboles (s). Une valeur *vrai* pour $X_{i,j,k}$ signifiant que le symbole s se trouve aux coordonnées i,j du tableau. La structure de ce tableau peut être décrite à l'aide d'un ensemble fini de clauses booléennes sur les variables $X_{i,j,s}$ (certaines pour spécifier que chaque case du tableau est remplie d'un seul symbole, d'autres pour spécifier que le passage d'une ligne à la suivante correspond à une transition de M, et enfin d'autres pour définir la configuration initiale et préciser la configuration finale). L'existence d'un tableau vérifiant ces règles (et donc l'acceptation de la donnée) se traduit donc par une instance du problème SAT. La partie la plus difficile (et la plus importante !) de la démonstration est de prouver que la taille de l'instance du problème SAT obtenue croît seulement polynomialement en fonction de la taille de la donnée du problème initial. □

4.12. — Exemples. Voici d'autres exemples de problèmes **NP**-Complets.

• L'existence d'un chemin hamiltonien dans un graphe (un chemin qui passe une fois et une seule par chaque sommet).

• Le problème de la somme partielle (*subset-sum problem*). Etant donné un ensemble fini $A = \{a_1, \ldots, a_n\}$ d'entiers positifs et b un autre entier positif. Existe-t-il une somme $\sum_{i\in I} a_i$ de certains des a_i ($I \subseteq 1, \ldots, n$), égale à b ?

5. Complexité des algorithmes probabilistes

Coûts

Les algorithmes considérés jusqu'à présent sont déterministes : leur déroulement ne dépend que de la valeur d'entrée. Mais de nombreux algorithmes utiles font intervenir le hasard au cours de leur déroulement. Ils sont modélisés par les machines de Turing probabilistes. Puisque leur temps d'exécution pour une même donnée peut varier, on a un nouveau type de coût (en temps ou en espace) intéressant pour un tel algorithme. Soit $\mathcal{C}'_{\text{temps}}(x)$ le coût moyen de l'algorithme pour une donnée x (la moyenne étant faite sur toutes les séquences de choix possibles). Le coût probabiliste (*expected running time*) de l'algorithme est

$$\mathcal{C}^{\text{exp}}_{\text{temps}}(n) = \max_x \mathcal{C}'_{\text{temps}}(x) \quad x \text{ parcourant l'ensemble des données de longueur} \leqslant n.$$

Il est aussi utile de préciser ce qu'on entend par coût maximum d'un algorithme probabiliste. Soit $\mathcal{C}''_{\text{temps}}(x)$ le coût maximum pour une donnée x (le nombre d'opérations nécessaires pour que l'algorithme se termine, quelle que soit la séquence de choix). Le coût maximum de l'algorithme est

$$\mathcal{C}^{\max}_{\text{temps}}(n) = \max_x \mathcal{C}''_{\text{temps}}(x) \quad x \text{ parcourant l'ensemble des données de longueur} \leqslant n.$$

Classes **ZPP**, **RP** et **BPP**

D'autre part, certains algorithmes probabilistes utiles ne calculent pas toujours la réponse exacte au problème de décision considéré et peuvent parfois calculer deux réponses différentes pour une même donnée, suivant la séquence de choix. La réponse qu'ils calculent dans ce cas est alors à interpréter sous la forme *probablement oui*, ou *probablement non*. Cette variété d'algorithmes probabilistes invite à définir d'autres classes de problèmes de décision.

5.1. — Définitions. • Un problème de décision appartient à *la classe* **ZPP** (*Zero-sided (error) Probabilistic Polynomial*) s'il existe une machine de Turing probabiliste qui le résout et qui s'arrête en temps **probabiliste** polynomial (que la donnée soit dans D^+ ou dans D^-).

• Un problème de décision appartient à *la classe* **RP** (*Random Polynomial*) s'il existe une machine de Turing probabiliste qui s'arrête toujours en un temps **maximum** polynomial et qui, lorsque la donnée est dans D^+, s'arrête dans l'état q_y avec probabilité $\geqslant 1/2$ et, lorsque la donnée est dans D^-, ne l'accepte pas.

• Un problème de décision appartient à *la classe* **BPP** (*Bounded (error) Probabilistic Polynomial*) s'il existe une machine de Turing probabiliste qui s'arrête toujours en un temps **maximum** polynomial et qui, lorsque la donnée est dans D^+, s'arrête dans l'état q_y avec probabilité $\geqslant 2/3$ et, lorsque la donnée est dans D^-, s'arrête dans l'état q_y avec probabilité $\leqslant 1/3$.

5.2. — Proposition. *On a les relations suivantes*

$$\mathbf{ZPP} = co\text{-}\mathbf{ZPP}, \qquad \mathbf{BPP} = co\text{-}\mathbf{BPP},$$
$$\mathbf{P} \subseteq \mathbf{ZPP} \subseteq \mathbf{RP} \subseteq \mathbf{BPP},$$
$$\mathbf{RP} \subseteq \mathbf{NP}, \qquad \mathbf{ZPP} = \mathbf{RP} \cap co\text{-}\mathbf{RP}.$$

Démonstration — Montrons $\mathbf{ZPP} \subseteq \mathbf{RP}$. Soit $\mathcal{D}$ un problème de décision de la classe **ZPP**. Il existe donc un polynôme P et une machine de Turing probabiliste M dont le temps de calcul probabiliste est majoré par P et résolvant $\mathcal{D}$. Pour une entrée fixée, le temps de calcul ne peut dépasser $2P$ qu'avec probabilité inférieure à $1/2$. On peut modifier la machine M en la forçant à s'arrêter dans un état autre que q_y dès qu'elle a effectué $2P$ pas. Cette machine modifiée montre que $\mathcal{D}$ appartient à **RP**. Les autres relations sont laissées à titre d'exercice. □

Un algorithme probabiliste qui montre qu'un problème de décision est dans l'une des classes ci-dessus est parfois appelé :

• algorithme *de Las Végas* pour la classe **ZPP**,

• algorithme *de Monté-Carlo* pour la classe **RP**,

• algorithme *d'Atlantic City* pour la classe **BPP**.

5.3. — Exercice. • Montrer que le problème de décision posant la question, pour $n \in \mathbb{N}^*$, "n est-il composé ?" est dans la classe **RP**. Indication : lire la section "Tests de primalité" du chapitre 14.

• Montrer que le problème de décision posant la question, pour p premier, "2 admet-il une racine carrée modulo p inférieure à $p/4$?" est dans la classe **ZPP**. Indication : lire la section "Racines carrées" du chapitre 8.

5.4. — Remarque. La plupart des spécialistes estiment que la classe **BPP** est une définition raisonnable de la classe des problèmes de décision résolubles efficacement en pratique, dans le cadre de la mécanique classique. La question de savoir si le cadre de la mécanique quantique permettrait d'agrandir la classe des problèmes résolubles efficacement reste ouverte.

Chapitre 8

Arithmétique pratique

L'essentiel de ce chapitre est présenté à la fin du livre de Menezes, van Oorschot & Vanstone [84]. Bien sûr, la référence ultime est Knuth [56].

1. L'anneau $\mathbb{Z}$

Opérations élémentaires

Nous avons l'habitude de représenter les nombres entiers en utilisant la base 10. L'immense majorité des calculateurs électroniques représentent leurs données en utilisant la base 2 (ou une puissance de 2), ne convertissant en base 10 que les résultats finaux avant de les présenter à l'écran. Toutefois, à part ce choix différent de la base, les algorithmes le plus souvent utilisés pour réaliser les opérations élémentaires sont essentiellement les mêmes que ceux que l'on apprend à l'école primaire pour calculer à la main. La longueur d'un entier positif n représenté sous forme binaire (i.e. le nombre de bits nécessaires) est $\lfloor \log_2 n \rfloor + 1 = \Theta(\ln n)$. On vérifie facilement que les complexités des quatre opérations élémentaires sur deux entiers a, b positifs et inférieurs à n ($a \leqslant b$) sont celles données par la table suivante.

Opération	Complexité
Addition $a + b$	$O(\ln a + \ln b) = O(\ln n)$
Soustraction $a - b$	$O(\ln a + \ln b) = O(\ln n)$
Multiplication ab	$O(\ln a \ln b) = O(\ln^2 n)$
Division euclidienne de a par b	$O\big((\ln a - \ln b)\ln b\big) = O(\ln^2 n)$

Table 1. Complexité des opérations élémentaires

1.1. — Cas particuliers. La multiplication par 2 peut se faire par un simple décalage et est donc d'un coût très faible. L'élévation au carré, si on utilise un algorithme judicieux, coûte moitié moins qu'une multiplication générique.

La manière la plus pratique pour traiter les nombres négatifs est la représentation par complément. Elle consiste à représenter chaque entier de l'intervalle $[n/2, n/2[$ (n étant une puissance de la base) par le développement de l'unique entier de l'intervalle $[0, n[$ qui lui soit congru modulo n. Les algorithmes d'addition et de soustraction sont inchangés, excepté la détection des débordements.

Pour de grands entiers, le coût de la multiplication peut être amélioré par l'utilisation de l'algorithme de Karatsuba & Ofman [55] qui est en $O(n^{\log_2 3})$ (on a $\log_2 3 \simeq 1.58$), ou même par les méthodes utilisant la transformée de Fourier discrète, qui ramènent à une complexité en $O(n \ln n \ln\ln n)$.

L'algorithme d'Euclide

L'anneau $\mathbb{Z}$ est principal. Le pgcd de deux entiers a et b est l'entier positif vérifiant $\mathrm{pgcd}(a,b)\mathbb{Z} = a\mathbb{Z} \cap b\mathbb{Z}$. L'algorithme d'Euclide permet de calculer le pgcd en utilisant l'égalité $\mathrm{pgcd}(a,b) = \mathrm{pgcd}(b, a \bmod b)$ pour $a, b \in \mathbb{N}^*$ tels que $a > b$:

Entrée : deux entiers positifs a et b (non tous deux nuls).
Sortie : $\mathrm{pgcd}(a,b)$.

Tant que $b \neq 0$ faire
    $(a,b) := (b, a \bmod b)$
Retourner a

Pour évaluer son coût, notons (a_i) la suite des restes avec $a_0 = a$, $a_1 = b$, …, $a_k = d$, $a_{k+1} = 0$ (en supposant $a \geqslant b$). Seuls les coûts des divisions euclidiennes sont non-négligeables et voici leur somme :

$$\begin{aligned}\sum_{i=1}^{k}(\ln a_{i-1} - \ln a_i)\ln a_i &= \sum_{i=0}^{k-1}\ln a_i \ln a_{i+1} - \sum_{i=1}^{k}\ln a_i \ln a_i \\ &= \ln a_0 \ln a_1 + \sum_{i=1}^{k-1}(\ln a_i \ln a_{i+1} - \ln a_i \ln a_i) - \ln a_k \ln a_k \\ &\leqslant \ln a_0 \ln a_1 = \ln a \ln b \qquad \text{puisque la suite des } a_i \text{ est décroissante.}\end{aligned}$$

Le coût de l'algorithme d'Euclide est donc quadratique.

La version étendue de l'algorithme d'Euclide permet en plus de calculer des coefficients de Bézout. Son coût est encore quadratique.

Entrée : deux entiers positifs a et b (non tous deux nuls).
Sortie : trois entiers u, v, d tels que $au + bv = d = \mathrm{pgcd}(a, b)$.
Variables entières : $u_1, v_1, u_2, v_2, u_3, v_3, q, r$.

Si $a = 0$ alors retourner $(0, 1, b)$ et arrêter l'algorithme
$--$ [$Au_1 + Bv_1 = a$ et $Au_2 + Bv_2 = b$, où A, B sont les valeurs initiales de a, b]
$(u_1, v_1) := (1, 0)$
$(u_2, v_2) := (0, 1)$
Tant que $b \neq 0$ faire
    $(q, r) :=$ quotient et reste de la division euclidienne de a par b
    $(u_3, v_3) := (u_1 - qu_2, v_1 - qv_2)$
    $(u_1, v_1, a) := (u_2, v_2, b)$
    $(u_2, v_2, b) := (u_3, v_3, r)$
Refaire
Retourner (u_1, v_1, a)

2. L'anneau $\mathbb{Z}/m\mathbb{Z}$

Opérations élémentaires

On représente chaque élément de $\mathbb{Z}/m\mathbb{Z}$ de la même façon que son unique représentant dans l'intervalle $[0, m-1]$. Les opérations modulaires élémentaires se déduisent des opérations élémentaires dans $\mathbb{Z}$:

• Pour additionner modulo m, il suffit d'additionner les représentants et, si le résultat dépasse m, de lui retrancher m. Le coût est donc en $O(\ln m)$.

• Pour soustraire modulo m, il suffit de soustraire les représentants et, si le résultat est négatif, de lui ajouter m. Le coût est encore en $O(\ln m)$.

• Pour multiplier modulo m, il suffit de multiplier les représentants et de réduire le résultat modulo m, c'est-à-dire calculer son reste dans la division euclidienne par m. Le coût est en $O(\ln^2 m)$.

• Pour calculer l'inverse modulo m d'un entier a, il suffit d'utiliser l'algorithme d'Euclide étendu. En effet, si a est inversible modulo m alors $\mathrm{pgcd}(a, m) = 1$ et on obtient des entiers u, v vérifiant $ua + vm = 1$. L'inverse de a sera donc représenté par le reste de la division euclidienne de u par m. Le coût est en $O(\ln^2 m)$.

Equation $ax \equiv b$ modulo m

Soient a, b, m des entiers, avec $m \geqslant 2$. On cherche à résoudre l'équation $ax \equiv b$ modulo m. Posons $d = \mathrm{pgcd}(a, m)$. Si $d \nmid b$ alors l'équation n'a pas de solution. Sinon, on peut poser $a_1 = a/d$, $b_1 = b/d$, $m_1 = m/d$ et l'équation d'origine est équivalente à $a_1 x \equiv b_1$ modulo m_1. Mais alors $\mathrm{pgcd}(a_1, m_1) = 1$ et on obtient $x \equiv a_1^{-1} b_1$ modulo m_1. Notons que, modulo m, il y a exactement d solutions.

2.1. — Exercice. Résoudre l'équation $6x \equiv 15$ modulo 33.

Le théorème chinois

Si $m_1, m_2, \ldots, m_k$ sont des entiers premiers entre eux deux-à-deux et si $m = \prod_{i=1}^{k} m_i$ alors on a un isomorphisme

$$\begin{aligned} \mathbb{Z}_m &\longrightarrow \mathbb{Z}_{m_1} \times \cdots \times \mathbb{Z}_{m_k} \\ x &\longmapsto (x \bmod m_1, \ldots, x \bmod m_k). \end{aligned}$$

Il se calcule explicitement à l'aide de k divisions euclidiennes. Calculer sa réciproque revient à trouver, étant données k valeurs $x_i \in \mathbb{Z}_{m_i}$ $(1 \leqslant i \leqslant k)$, un entier $x \in \mathbb{Z}_m$ vérifiant $x \equiv x_i$ modulo m_i pour chaque i. Cela est efficacement réalisé par l'algorithme de Garner :

Entrée : $x_1, \ldots, x_k$ tels que $x_i \in \mathbb{Z}_{m_i}$.
Sortie : $x \in \mathbb{Z}_m$ tel que $x \equiv x_i \pmod{m_i}$ pour $1 \leqslant i \leqslant k$.
Constantes pré-calculées : $C_i = \prod_{j=1}^{i-1} m_j^{-1} \bmod m_i$ pour $2 \leqslant i \leqslant k$.

$x := x_1$
Pour i de 2 à k faire
  $--$ $[x \equiv x_j$ pour $1 \leqslant j < i$ et $0 \leqslant x \leqslant \prod_{j=1}^{i-1} m_i - 1]$
  $x := x + \big((x_i - x)C_i \bmod m_i\big) \prod_{j=1}^{i-1} m_j$
Retourner x

2.2. — **Exercice.** Résoudre le système suivant, à l'aide de l'algorithme de Garner.

$$\begin{cases} 3x \equiv 2 \pmod{11} \\ 3x \equiv 4 \pmod{7} \\ 7x \equiv 5 \pmod{36} \end{cases}$$

2.3. — **Exercice.** Soient $a, b \in \mathbb{N}^*$ premiers entre eux. Soient $u = a^{-1} \bmod b$ et $v = b^{-1} \bmod a - a$. Montrer que $au + bv = 1$.

L'exponentiation

Etant donné un monoïde G pour lequel nous notons la loi multiplicativement, les deux algorithmes suivants permettent chacun de calculer a^e, pour $a \in G$ et $e \in \mathbb{N}$. On note e_k les chiffres binaires de e, c'est-à-dire $e = \sum_0^{d-1} e_k 2^k$. Leur coût est en $O(\ln e)$ instantiations de la loi de G. Par exemple, si ce monoïde est le groupe $(\mathbb{Z}/m\mathbb{Z})^*$, alors le coût global est en $O(\ln^2 m)(\ln e)$.

Entrée : $a \in G$, $e \in \mathbb{N}$.
Sortie : a^e.

$x := 1$
$b := a$
Tant que $e \neq 0$ faire
  $--$ $[b = a^{2^i}$ et $xa^{2^i e} = a^E$ à la i-ème étape$]$
  si e est impair alors $x := b * x$ finsi
  $b := b * b$
  $e := \lfloor e/2 \rfloor$
Refaire
Retourner x

Entrée et sortie : idem.

$x := 1$
$k := d$ (nombre de chiffres binaires de e)
Tant que $k \neq 0$ faire
  $--$ $[x = a^f$ avec $f = \sum_{j=0}^{d-k-1} e_{k+j} 2^j]$
  $x := x * x$
  si $e_{k-1} = 1$ alors $x := x * a$ finsi
  $k := k - 1$
Refaire
Retourner x

La représentation de Montgomery

Les opérations modulo m décrites ci-dessus s'obtiennent par une opération sur $\mathbb{Z}$ suivie d'une réduction. Le coût de cette réduction pèse lourd, en particulier pour la multiplication modulaire qui se termine par une division euclidienne par m très coûteuse. La représentation de Montgomery [88] permet de réduire ce coût en remplaçant la division par m par des divisions par un entier $R > m$ dont le coût est négligeable, par exemple si R est une puissance de la base utilisée.

On choisit donc un entier $R > m$ vérifiant $\mathrm{pgcd}(R, m) = 1$ tel que la division euclidienne par R soit facile et on détermine $m' = -m^{-1} \bmod R$. Chaque élément de $\mathbb{Z}/m\mathbb{Z}$, au lieu d'être représenté par son reste x modulo m, sera représenté par $\tilde{x} = xR \bmod m$. L'algorithme d'addition est inchangé puisque $\tilde{x} + \tilde{y} \bmod m = \widetilde{x + y}$. De même pour la soustraction. Par contre, on a $\tilde{x}\tilde{y} \equiv R\widetilde{xy}$ modulo m. Il faudra donc diviser modulo m le résultat de la multiplication $\tilde{x}\tilde{y}$ par R pour obtenir la représentation de Montgomery de xy. Voici l'algorithme de réduction de Montgomery qui permet de faire cette division à coût faible.

Entrée : Un entier T tel que $0 \leqslant T < Rm$.
Sortie : $TR^{-1} \bmod m$.

$u := (T \bmod R)m' \bmod R$
$t := (T + um)/R$
Si $t \geqslant m$ alors retourner $t - m$, sinon retourner t

La division par R dans le calcul de t est exacte. Si on néglige son coût ainsi que celui des deux réductions modulo R, il reste celui d'une multiplication et d'une ou deux additions modulo m.

Le sur-coût dû aux changements de représentations est négligeable s'il se fait seulement au début et à la fin d'un long calcul. Il peut être absorbé même pour une simple exponentiation si l'exposant est suffisamment grand.

3. Calcul du symbole de Jacobi

Pour $m, n \in \mathbb{Z}$ avec $n > 0$ impair, l'algorithme suivant (qui ressemble beaucoup à l'algorithme d'Euclide) calcule le symbole de Jacobi (m/n).

Entrée : m,n avec $n > 0$ impair.
Sortie : le symbole de Jacobi (m/n).

```
m := m mod n
j := 1
Tant que m > 1 faire
      si m est pair alors
            changer le signe de j si n ≡ 3 ou 5 modulo 8
            m := m/2
      sinon
            changer le signe de j si m ≡ n ≡ 3 mod 4
            (m, n) := (n mod m, m)
si m = 0 alors retourner 0 sinon retourner j
```

Par une analyse similaire à celle de l'algorithme d'Euclide, on voit que sa complexité est en $O(\ln(m)\ln(n))$.

4. Corps finis

Le livre de Lidl & Niederreiter [71] est une référence complète. Dans [83], on a des aspects algorithmiques.

Rappels

4.1. — Théorème. *Pour chaque puissance p^r d'un nombre premier, il existe un corps fini, unique à isomorphisme près, de cardinal p^r. Nous le noterons $\mathbb{F}_{p^r}$.*

Le corps fini $\mathbb{F}_{p^r}$ est de caractéristique p. On peut le construire en choisissant un polynôme f de degré r irréductible sur $\mathbb{Z}/p\mathbb{Z}$ (il en existe toujours). L'anneau quotient $\mathbb{Z}/p\mathbb{Z}[X]/(f)$ est alors un corps fini à p^r éléments. C'est un espace vectoriel sur le corps $\mathbb{F}_p = \mathbb{Z}/p\mathbb{Z}$ et la famille $\{1, X, \ldots, X^{n-1}\}$ en est une base.

4.2. — Théorème. *Pour chaque corps fini $\mathbb{F}_{p^r}$, le groupe multiplicatif $\mathbb{F}_{p^r}^*$ est cyclique.*

On se propose de calculer dans le corps fini $\mathbb{F}_q$, avec $q = p^r$. On suppose que l'on a une méthode de représentation des éléments de $\mathbb{F}_p$ et de l'arithmétique correspondante (c'est simplement de l'arithmétique modulo p).

Représentation polynomiale

On se fixe un polynôme f irréductible sur $\mathbb{F}_p$ et de degré r (il en existe). On a $\mathbb{F}_q \simeq \mathbb{F}_p[X]/(f)$ ce qui permet de représenter chaque élément de $\mathbb{F}_q$ par un polynôme $u(X)$ sur $\mathbb{F}_p$ de degré $< r$. De façon équivalente, en désignant par α une racine de f (dans $\mathbb{F}_q$), chaque élément de $\mathbb{F}_q$ s'écrit sous la forme $u(\alpha)$ où u est un polynôme de degré $< r$ (le même que ci-dessus). Cette représentation revient aussi à décomposer les éléments du $\mathbb{F}_p$-espace vectoriel $\mathbb{F}_q$ dans la base $1, \alpha, \ldots, \alpha^{r-1}$.

• L'addition se fait coordonnées par coordonnées (la soustration aussi).

• La multiplication. Multiplier les deux polynômes et réduire (au fur et à mesure) modulo f.

Entrée : deux éléments u, v de $\mathbb{F}_q$ représentés par leurs coordonnées.
Sortie : le produit uv représenté de la même façon.
Variables : vecteur de coordonnées w.

$w \leftarrow v_0 u$ −− [boucle sur les composantes de u et w]
Pour i de 1 à $r-1$ faire
  $u \leftarrow Xu \bmod f$ −− [décalage à gauche et réduction]
  $w \leftarrow w + v_i u$ −− [boucle]
Fin Pour
Retourner w

Si r est petit, on peut précalculer les coordonnées dans la base $1, \alpha, \ldots, \alpha^{r-1}$ des α^{i+j} pour $0 \leqslant i+j < 2r-2$ et obtenir le produit de deux éléments quelconques par bilinéarité.

Il existe (voir [60]) une version polynomiale de la représentation de Montgomery, qui permet d'accélérer les multiplications et donc exponentiations dans les corps finis de caractéristique 2.

• Le calcul d'inverse s'obtient par l'algorithme étendu d'Euclide (dans l'anneau euclidien $\mathbb{F}_p[X]$) ou bien par la formule $1/x = x^{q-2}$ (pour $x \neq 0$).

Représentation dite exponentielle (logarithmique en fait)

Choisir un générateur α de $\mathbb{F}_q^*$ (c'est à dire prendre pour f un polynôme primitif). On représente chaque élément non nul par son logarithme en base α. Traiter 0 séparément (e.g. poser $\log_g 0 = -1$).

• Le produit dans $\mathbb{F}_q$ se réduit à une addition modulo $q-1$: $\alpha^i \alpha^j = \alpha^{i+j \bmod q-1}$.

• L'exponentiation est aussi très simple : $(\alpha^i)^e = \alpha^{ie \bmod q-1}$.

• L'addition devient compliquée. Si q est petit, on peut précalculer les logarithmes de Zech z_i définis par $\alpha^{z_i} = 1 + \alpha^i$. On a alors, en supposant $i \leqslant j$, $\alpha^i + \alpha^j = \alpha^i(1 + \alpha^{j-i})$.

Cette représentation est très avantageuse pour q vraiment petit (< 100 ?). Elle est inutilisable pour q grand.

Bases normales.

Prendre comme base du $\mathbb{F}_p$-espace vectoriel $\mathbb{F}_q$ une famille de la forme $\alpha, \alpha^p, \ldots, \alpha^{p^{r-1}}$ (il existe toujours de telles bases).

• L'addition est celle des vecteurs : terme à terme. En caractéristique 2, c'est donc simplement le XOR bit à bit.

• La multiplication peut s'obtenir par bilinéarité après avoir tabulé les $\alpha^{p^i+p^j}$ pour $0 \leqslant i, j < r-1$. En fait, si on écrit

$$\alpha^{p^i+p^j} = \sum_{k=0}^{r-1} \lambda_{i,j,k} \alpha^{p^k},$$

alors on a, pour tout entier t, les sommes d'indices étant réduites modulo r,

$$\sum_{k=0}^{r-1} \lambda_{i+t,j+t,k} \alpha^{p^k} = \alpha^{p^{i+t}+p^{j+t}} = (\alpha^{p^i+p^j})^{p^t} = \sum_{k=0}^{r-1} \lambda_{i,j,k} \alpha^{p^{k+t}} = \sum_{k=0}^{r-1} \lambda_{i,j,k-t} \alpha^{p^k}.$$

Donc, $\lambda_{i+t,j+t,k} = \lambda_{i,j,k-t}$. En particulier, $\lambda_{i,j,k} = \lambda_{i-k,j-k,0}$.

• Pour l'exponentiation, on peut décomposer l'exposant en base p. L'élévation à la puissance p est une rotation des coordonnées. C'est particulièrement pratique en base 2, où on se ramène de cette manière à au plus $r-1$ multiplications (on peut supposer que l'exposant est $\leqslant q-2$).

Cas de la caractéristique 2

Dans toutes les représentations (sauf exponentielle), un élément de $\mathbb{F}_{2^r}$ est représenté par un train binaire de longueur r.

• L'addition est le XOR bit à bit.

• L'élévation au carré est une rotation des coordonnées.

• Le calcul d'un reste modulo f (et du quotient correspondant) peut se faire par un registre à décalage effectuant une division par les puissances décroissantes, dont voici un exemple :

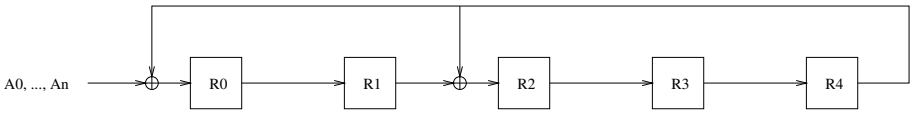

Figure 2. Calcul de reste modulo $X^5 + X^2 + 1$

Les registres R_0 à R_4 contiennent initialement 0. Les coefficients du polynôme à diviser sont introduits un à un dans le circuit en commençant par le coefficient dominant. Le reste est représenté par le contenu des registres après fonctionnement du circuit. Les quotient est représenté par les coefficients successifs sortant du registre R_4.

4.3. — Exercice. Construire un circuit effectuant une division par les puissances croissantes.

• On peut aussi effectuer une multiplication par un élément constant à l'aide d'un registre à décalage :

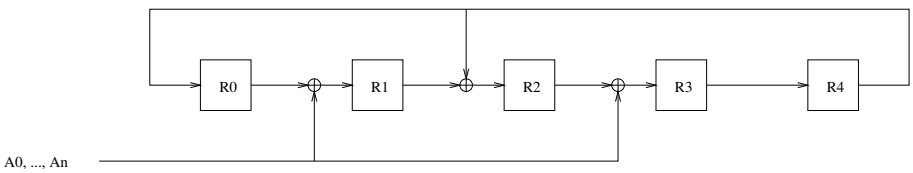

Figure 3. Multiplication par $X^3 + X$ modulo $X^5 + X^2 + 1$

4.4. — Exercice. Soit $q = 2^{155}$. Montrer qu'on peut réaliser les inversions ($x \mapsto x^{-1}$) dans $\mathbb{F}_q$ au coût de dix multiplications. Indications : utiliser une base normale et remarquer que

$$\begin{aligned} 2^{155} - 2 &= 2(2^{77} - 1)(2^{77} + 1), \\ 2^{77} - 1 &= 2(2^{19} - 1)(2^{19} + 1)(2^{38} + 1) + 1, \\ 2^{19} - 1 &= 2(2^9 - 1)(2^9 + 1) + 1, \\ 2^9 - 1 &= 2(2 + 1)(2^2 + 1)(2^4 + 1) + 1. \end{aligned}$$

5. Racines carrées

L'algorithme suivant (Tonelli & Shanks) calcule une racine carrée (si elle existe) d'un élément non nul d'un corps fini $\mathbb{F}_q$ (avec q puissance d'un nombre premier). Cas particulier : appliqué au corps $\mathbb{F}_p$ avec p premier, cet algorithme calcule des racines carrées modulo p. Sa complexité moyenne est en $O(\ln^4(q))$.

Entrée : $a \in \mathbb{F}_q$ non nul.
Sortie : une racine carrée de a, ou erreur si a n'est pas carré dans $\mathbb{F}_q$.

Trouver k et r tels que $q-1 = 2^k r$ avec r impair
$x := a^{(r+1)/2}$
$b := a^r$
Parcourir les éléments de $\mathbb{F}_q^*$ jusqu'à trouver g vérifiant $g^{(q-1)/2} = -1$ (i.e. g non carré)
$g := g^r$
Tant que $b \neq 1$ faire
    $--$ [$ab = x^2$ et l'ordre de b est une puissance de 2]
    Rechercher m tel que $2^m =$ ordre de b
    Si $m = k$ alors erreur (a n'est pas un carré dans $\mathbb{F}_q$)
    $t := g^{2^{k-m-1}}$ $--$ [t^2 est d'ordre 2^m]
    $b := bt^2$
    $x := tx$
Retourner x

5.1. — **Remarques.** • Si q est pair, chaque $a \in \mathbb{F}_q^*$ admet une unique racine carrée $a^{q/2}$.

• Si $q \equiv 3$ modulo 4 et $a \neq 0$ est un carré dans $\mathbb{F}_q$ (donc $a^{(q-1)/2} = 1$), les racines carrées de a sont $\pm a^{(q+1)/4}$.

5.2. — **Remarque.** Grâce au théorème chinois et à l'algorithme de Garner, il est possible de calculer des racines carrées modulo un entier n dès que l'on connaît sa factorisation. Inversement, si on dispose d'un algorithme efficace pour le calcul de racines carrées modulo un nombre composé n, alors on peut facilement factoriser n. En effet, on peut supposer que n possède au moins deux facteurs premiers distincts (car il est assez facile de factoriser une puissance parfaite). Dans ce cas, chaque carré possède au moins quatre racines carrées. En choisissant un entier $x \in \mathbb{Z}_n$ au hasard et en utilisant l'algorithme pour calculer une racine carrée de x^2 modulo n, on a au moins une chance sur deux d'obtenir un nombre y distinct de $\pm x$ et tel que $x^2 \equiv y^2$ modulo n. Mais alors $\mathrm{pgcd}(x-y, n)$ est un facteur non trivial de n.

Chapitre 9

Générateurs aléatoires

La cryptographie a besoin de nombres aléatoires. Par exemple pour générer des clés (que ce soit pour faire du chiffrement à flot ou bien en cryptographie publique) et aussi pour les défis dans les protocoles d'identification. Si les valeurs choisies sont pas aléatoires, alors la sécurité du protocole correspondant peut être gravement altérée. Pour produire des nombres aléatoires, on a besoin de savoir générer des suites de bits qui non seulement possèdent des propriétés statistiques compatibles avec des suites aléatoires, mais aussi ne puissent pas, en temps raisonnable, être distinguées de suite aléatoires vraies.

1. Générateurs cryptographiquement sûrs

Ici $k \in \mathbb{N}$ est un paramètre, dit *paramètre de sécurité* et la plupart de ce qui suit n'a de sens que lorsqu'on fait varier k (et ℓ). Une fonction *négligeable* est une fonction de $\mathbb{N}$ dans $\mathbb{R}^+$ qui décroît plus vite que l'inverse de toute fonction polynomiale.

1.1. — Définition. Soit $\ell \in \mathbb{N}$ une fonction de k telle que $k < \ell$. Un (k,ℓ)-*générateur aléatoire* est une fonction F calculable en temps polynomial en ℓ de $\mathcal{A} \subseteq \{0,1\}^k$ dans $\{0,1\}^\ell$.

En composant la distribution uniforme sur $\mathcal{A}$ avec F, on obtient une distribution sur $\{0,1\}^\ell$ appelée la *distribution induite* par F.

Pour des applications cryptographiques, on souhaite que les ℓ bits fournis par un générateur (ou plus exactement la distribution induite sur $\{0,1\}^\ell$) ne vérifient aucune propriété qui pourrait les différencier de bits aléatoires (la distribution uniforme sur $\{0,1\}^\ell$). Pour cela, on introduit la notion suivante.

1.2. — Définition. Soit F un (k,ℓ)-générateur aléatoire et $\epsilon > 0$. Un ϵ-*distingueur pour* F est un algorithme D probabiliste polynomial $\{0,1\}^\ell \to \{0,1\}$ tel que

$$\mathrm{prob}_F\{D(r_1,\ldots,r_\ell) = 1\} - \mathrm{prob}\{D(r_1,\ldots,r_\ell) = 1\} \geqslant \epsilon$$

où prob_F signifie que les bits $r_1,\ldots,r_\ell$ sont choisis selon la distribution induite par F et prob qu'ils sont choisis selon la distribution uniforme.

1.3. — Définition. Un (k,ℓ)-générateur aléatoire f est dit *sûr* (au sens cryptographique) s'il n'existe d'ϵ-distingueur pour F que pour des fonctions ϵ négligeables (par rapport au paramètre ℓ).

Extrapoleurs

1.4. — Définition. Soient F un (k,ℓ)-générateur aléatoire, $t \in [1,\ell]$ un entier, et $\epsilon > 0$. On note $(r_1,\ldots,r_\ell)$ des bits choisis suivant la distribution induite par F sur $\{0,1\}^\ell$. Un *extrapoleur d'ordre t pour F d'avantage ϵ* est un algorithme probabiliste polynomial calculant le bit r_t en fonction des précédents $r_1,\ldots,r_{t-1}$ avec probabilité $\geqslant 1/2 + \epsilon$.

A partir d'un extrapoleur E d'ordre t et d'avantage ϵ pour F, on obtient un ϵ-distingueur D pour F de la manière suivante.

Entrée : $(r_1,\ldots,r_\ell) \in \{0,1\}^\ell$.
Sortie : un élément de $\{0,1\}$.

Si $E(r_1,\ldots,r_{t-1}) = r_t$
  alors retourner 1
  sinon retourner 0

En effet, la probabilité que $E(r_1,\ldots,r_{t-1})$ soit égal à r_t vaut au moins $1/2+\epsilon$ lorsque $(r_1,\ldots,r_\ell)$ est choisi selon la distribution induite par F et vaut $1/2$ lorsque $(r_1,\ldots,r_\ell)$ est choisi selon la distribution uniforme.

Le résultat suivant exprime que les distingueurs construits de cette façon sont universels, c'est-à-dire qu'ils suffisent pour déterminer si un générateur aléatoire est sûr.

1.5. — Théorème (Yao). *Un (k,ℓ)-générateur aléatoire F est sûr si et seulement si, pour chaque $t \in [1,\ell]$ entier, il n'existe pas d'extrapoleur pour F, autres que d'avantage négligeable.*

DÉMONSTRATION — D'après la construction ci-dessus de distingueur à partir d'un extrapoleur, F ne peut être sûr que si tout extrapoleur n'est que d'avantage négligeable.

Inversement, supposons que F ne soit pas sûr. Pour $0 \leqslant i \leqslant \ell$, notons F_i la distribution sur $\{0,1\}^\ell$ selon laquelle les i premiers bits $r_1,\dots,r_i$ sont produits par F et les autres $r_{i+1}\dots,r_\ell$ sont choisis uniformément au hasard. Ainsi, F_0 est la distribution uniforme et F_ℓ celle induite par F. Par hypothèse, il existe un ϵ-distingueur D pour F telle que ϵ ne soit pas négligeable, c'est-à-dire

$$\operatorname{prob}_{F_l}\{D(r_1,\dots,r_\ell)=1\} - \operatorname{prob}_{F_0}\{D(r_1,\dots,r_\ell)=1\} \geqslant \epsilon.$$

Donc il existe un $t \in [1..\ell]$ et une fonction ϵ' pas négligeable tels que

$$\operatorname{prob}_{F_t}\{D(r_1,\dots,r_\ell)=1\} - \operatorname{prob}_{F_{t-1}}\{D(r_1,\dots,r_\ell)=1\} \geqslant \epsilon' \tag{1}$$

Considérons alors l'algorithme E_t suivant.

Entrée : $(r_1,\dots,r_{t-1}) \in \{0,1\}^{t-1}$.
Sortie : élément de $\{0,1\}$.

choisir $r_t,\dots,r_\ell$ au hasard
Si $D(r_1,\dots,r_\ell)=1$
  alors retourner r_t
  sinon retourner $\overline{r_t}$

Nous allons montrer que E_t est un extrapoleur d'avantage ϵ' pour F. Tout d'abord, on a

$$\operatorname{prob}_{F_{t-1}}\{r_1,\dots,r_{t-1}\} = 2^{\ell-t+1}\operatorname{prob}_{F_{t-1}}\{r_1,\dots,r_\ell\} \quad\text{et}\quad \operatorname{prob}_{F_t}\{r_1,\dots,r_t\} = 2^{\ell-t}\operatorname{prob}_{F_t}\{r_1,\dots,r_\ell\}.$$

Mais, $\operatorname{prob}_{F_{t-1}}\{r_1,\dots,r_{t-1}\}\operatorname{prob}_F\{r_t \mid r_1,\dots,r_{t-1}\} = \operatorname{prob}_{F_t}\{r_1,\dots,r_t\}$, donc

$$\operatorname{prob}_{F_{t-1}}\{r_1,\dots,r_\ell\}\operatorname{prob}_F\{r_t \mid r_1,\dots,r_{t-1}\} = \frac{1}{2}\operatorname{prob}_{F_t}\{r_1,\dots,r_\ell\}. \tag{2}$$

Ensuite, pour alléger les notations, nous poserons $\boldsymbol{r} = (r_1,\dots,r_\ell)$ et $\boldsymbol{r}_i = (r_1,\dots,r_i)$. La probabilité que E_t prévoie le bit r_t correctement est donnée par

$$\begin{aligned}
&\operatorname{prob}_F\{E_t(\boldsymbol{r}_{t-1}) = r_t\}\\
&\quad= \sum_{\boldsymbol{r}} \operatorname{prob}_{F_{t-1}}\{\boldsymbol{r}\}\big(\operatorname{prob}_F\{r_t \mid \boldsymbol{r}_{t-1}\}\operatorname{prob}\{D(\boldsymbol{r})=1\} + \operatorname{prob}_F\{\overline{r_t} \mid \boldsymbol{r}_{t-1}\}\operatorname{prob}\{D(\boldsymbol{r})=0\}\big)\\
&\quad= \sum_{\boldsymbol{r}} \operatorname{prob}_{F_{t-1}}\{\boldsymbol{r}\}\big(\operatorname{prob}_F\{r_t \mid \boldsymbol{r}_{t-1}\}\operatorname{prob}\{D(\boldsymbol{r})=1\} + (1-\operatorname{prob}_F\{r_t \mid \boldsymbol{r}_{t-1}\})\operatorname{prob}\{D(\boldsymbol{r})=0\}\big)\\
&\quad= \frac{1}{2}\sum_{\boldsymbol{r}} \operatorname{prob}_{F_t}\{\boldsymbol{r}\}\big(\operatorname{prob}\{D(\boldsymbol{r})=1\} - \operatorname{prob}\{D(\boldsymbol{r})=0\}\big) + \sum_{\boldsymbol{r}} \operatorname{prob}_{F_{t-1}}\{\boldsymbol{r}\}\operatorname{prob}\{D(\boldsymbol{r})=0\} \quad \text{par (2)}\\
&\quad= \frac{1}{2}\sum_{\boldsymbol{r}} \operatorname{prob}_{F_t}\{\boldsymbol{r}\}\big(2\operatorname{prob}\{D(\boldsymbol{r})=1\} - 1\big) + \sum_{\boldsymbol{r}} \operatorname{prob}_{F_{t-1}}\{\boldsymbol{r}\}\big(1-\operatorname{prob}\{D(\boldsymbol{r})=1\}\big)\\
&\quad= \sum_{\boldsymbol{r}} \operatorname{prob}_{F_t}\{\boldsymbol{r}\}\operatorname{prob}\{D(\boldsymbol{r})=1\} - \frac{1}{2} + 1 - \sum_{\boldsymbol{r}} \operatorname{prob}_{F_{t-1}}\{\boldsymbol{r}\}\operatorname{prob}\{D(\boldsymbol{r})=1\}\\
&\quad= \frac{1}{2} + \sum_{\boldsymbol{r}} \big(\operatorname{prob}_{F_t}\{\boldsymbol{r}\} - \operatorname{prob}_{F_{t-1}}\{\boldsymbol{r}\}\big)\operatorname{prob}\{D(\boldsymbol{r})=1\}\\
&\quad\geqslant \frac{1}{2} + \epsilon' \qquad \text{d'après (1).}
\end{aligned}$$

Ceci termine la démonstration. □

2. Le générateur BBS

Soit $N = pq$ en entier de Blum (définition 5.3.15) tel que $2^{k-1} < N \leqslant 2^k$ et soient

$$\mathbb{Z}_N^+ = \{x \in \mathbb{Z}_N \mid (x/N) = 1\}$$
$$Q = \{x^2 \bmod N \mid x \in \mathbb{Z}_N,\ \mathrm{pgcd}(x, N) = 1\} \subseteq \mathbb{Z}_N^+.$$

L'application $s : x \mapsto x^2 \bmod N$ est une bijection sur Q (voir proposition 5.3.18).

On identifie $\mathbb{Z}_N^+$ à une partie $\mathcal{A}$ de $\{0,1\}^k$, par l'écriture binaire naturelle des entiers. Soit $\ell > k$ une fonction polynomiale de k. Pour $a_0 \in \mathcal{A}$, on pose

$$a_i = s(a_{i-1}) \quad \text{et} \quad r_i = a_i \bmod 2, \qquad \text{pour } 1 \leqslant i \leqslant \ell.$$

C'est un (k, ℓ)-générateur, appelé le *générateur BBS* (Blum-Blum-Shub).

Sécurité de BBS

2.1. — **Lemme.** *Soit F un (k, ℓ)-générateur aléatoire et notons $r_1, \ldots, r_\ell$ les bits générés par F. Le générateur aléatoire générant les mêmes bits mais dans l'ordre inverse $r_\ell, \ldots, r_1$ est sûr si et seulement si F est sûr.*

Démonstration — Exercice. □

2.2. — **Lemme.** *Un (k, ℓ)-générateur aléatoire F est sûr si et seulement si, pour chaque $u \in [0, \ell - 1]$, il n'existe pas d'extrapoleur de bit* ***précédent*** *:*

$$E : (r_{\ell-u+1}, \ldots, r_\ell) \longmapsto r_{\ell-u}.$$

autres que d'avantage négligeable.

Démonstration — Appliquer le théorème de Yao et le lemme précédent. □

2.3. — **Lemme.** *Si le générateur BBS n'est pas sûr alors on a un algorithme qui reconnait dans $\mathbb{Z}_N^+$ les éléments de Q avec avantage pas négligeable.*

Démonstration — Si BBS n'est pas sûr alors, d'après le lemme précédent, il existe un $u \in [0, \ell - 1]$ et un extrapoleur E de bit **précédent** d'avantage ϵ pas négligeable. Comme s est une permutation de Q, l'algorithme E garde le même avantage ϵ si on l'applique aux premiers bits générés :

$$E : (r_1, \ldots, r_u) \longmapsto (\pm a_0 \bmod N) \bmod 2,$$

le signe valide étant celui tel que $\pm a_0 \bmod N \in Q$. Considérons alors l'algorithme B suivant.

Entrée : $a \in \mathbb{Z}_N^+$.
Sortie : Elément de $\{0,1\}$ (1 pour $a \in Q$, 0 sinon).

$a_0 \leftarrow a$
Pour i de 1 à ℓ, calculer $a_i = a_{i-1}^2 \bmod N$ et $r_i = a_i \bmod 2$
$r_0 \leftarrow E(r_1, \ldots, r_u)$
Si $r_0 = a \bmod 2$ alors retourner 1 sinon retourner 0.

Pour $a \in \mathbb{Z}_n^+$, cet algorithme détermine si $a \in Q$ avec avantage ϵ. □

2.4. — Lemme. *Soit A un algorithme de décision d'avantage ϵ, pour lequel la probabilité d'erreur est indépendante de l'entrée. En itérant $2m+1$ fois l'algorithme A et en retenant la décision majoritaire, on obtient un algorithme dont la probabilité d'erreur est $\leqslant (1-4\epsilon^2)^m/2$.*

DÉMONSTRATION — Pour $0 \leqslant i \leqslant 2m+1$, la probabilité qu'exactement i réponses de A soient correctes est

$$C_i^{2m+1}\left(\frac{1}{2}+\epsilon\right)^i\left(\frac{1}{2}-\epsilon\right)^{2m+1-i}.$$

La probabilité P que la décision majoritaire soit fausse vérifie donc

$$\begin{aligned}
P &\leqslant \sum_{i=0}^{m} C_{2m+1}^i\left(\frac{1}{2}+\epsilon\right)^i\left(\frac{1}{2}-\epsilon\right)^{2m+1-i}\\
&= \left(\frac{1}{2}+\epsilon\right)^m\left(\frac{1}{2}+\epsilon\right)^{m+1}\sum_{i=0}^{m} C_{2m+1}^i\left(\frac{1}{2}+\epsilon\right)^{i-m}\left(\frac{1}{2}-\epsilon\right)^{m-i}\\
&= \left(\frac{1}{2}+\epsilon\right)^m\left(\frac{1}{2}-\epsilon\right)^{m+1}\sum_{i=0}^{m} C_{2m+1}^i\left(\frac{1/2+\epsilon}{1/2-\epsilon}\right)^{m-i}\\
&\leqslant \left(\frac{1}{2}+\epsilon\right)^m\left(\frac{1}{2}-\epsilon\right)^{m+1}\sum_{i=0}^{m} C_{2m+1}^i\\
&= 4^m\left(\frac{1}{2}+\epsilon\right)^m\left(\frac{1}{2}-\epsilon\right)^{m+1}\\
&\leqslant \frac{(1-4\epsilon^2)^m}{2}
\end{aligned}$$

ce qui est la majoration annoncée. □

2.5. — Théorème. *Le générateur BBS est sûr sous l'hypothèse que le problème de la résidualité quadratique modulo N est difficile.*

DÉMONSTRATION — Supposons que le générateur BBS ne soit pas sûr. Considérons l'algorithme B dont l'existence est assurée par le lemme 2.3. Pour δ arbitrairement faible, on va appliquer plusieurs fois l'algorithme B pour obtenir un algorithme C de taux d'erreur inférieur à δ :

Entrée : $a \in \mathbb{Z}_N^+$.
Sortie : Elément de $\{0,1\}$ (1 pour $a \in Q$, 0 sinon).

$c \leftarrow 0$
Pour i de 1 à $2m+1$ faire
    choisir $r \in \mathbb{Z}_N$ relativement premier à N, au hasard
    choisir $\eta = \pm 1$ au hasard
    $b \leftarrow B(\eta r^2 a \bmod N)$
    Si $b = 1$ et $\eta = 1$ ou $b = 0$ et $\eta = -1$ alors $c \leftarrow c+1$
Si $c > m$ alors retourner 1 sinon retourner 0. −− [décision majoritaire]

La probabilité d'erreur de cet algorithme est $\leqslant (1-4\epsilon^2)^m/2$, comme le montre le lemme précédent. Soit δ tel que $0 < \delta < 1/2-\epsilon$, la probabilité d'erreur de C est $\leqslant \delta$ dès que $m \geqslant \ln(2\delta)/\ln(1-4\epsilon^2)$.

Cet algorithme contredit l'hypothèse selon laquelle le problème de la résidualité quadratique 5.3.19 est difficile. □

2.6. — Exercice. On suppose que 2 est un carré modulo N (i.e. $p \equiv q \equiv 7$ modulo 8). • Pour $x \in \mathbb{Z}_n$, on pose

$$m(x) = \begin{cases} 0 & \text{si } x < N/2,\\ 1 & \text{si } x > N/2. \end{cases}$$

Montrer qu'un oracle fournissant, à partir de $A \in Q$, le bit $m(a)$ où $a \in Q$ vérifie $s(a) = A$, permet d'inverser la fonction s (s'inspirer du théorème 10.1.9).

• En déduire qu'un oracle fournissant le bit $a \bmod 2$ au lieu de $m(a)$ permet aussi d'inverser la fonction s.

• Montrer que l'un ou l'autre oracle permettent de factoriser N en temps probabiliste polynomial.

Variante

Soit $R = \{x \in \mathbb{Z}_N^+ \mid x < N/2\}$. L'application $\mathbb{Z}_N \mapsto \mathbb{Z}_N$

$$s' : x \longmapsto |x^2 \bmod N|_N \quad \text{où } |y|_N \text{ désigne } \min(y, N-y)$$

induit une permutation sur R (exercice). On obtient donc une variante du générateur BBS en remplacant l'application s par s' et en choisissant a_0 relativement premier à N et dans l'intervalle $[0, N/2]$. Cette variante reste sûre sous une hypothèse un peu plus faible :

2.7. — Théorème. *Le générateur ainsi obtenu est sûr sous l'hypothèse que la factorisation de N est difficile.*

DÉMONSTRATION — Il s'agit de remplacer l'algorithme B du lemme 2.3 par l'algorithme que voici :

Entrée : $A \in R$.
Sortie : $a \bmod 2$, où $a \in R$ vérifie $s'(a) = A$.

$a_1 \leftarrow A$
$r_1 \leftarrow a_1 \bmod 2$
Pour i de 2 à l, calculer $a_i \leftarrow s'(a_{i-1})$ et $r_i \leftarrow a_i \bmod 2$
Retourner $E(r_1, \ldots, r_u)$

qui calcule la parité de la (bonne) racine carrée de A avec avantage ϵ. Il reste alors à appliquer le théorème suivant. □

2.8. — Théorème (Chor/Goldreich). *Un oracle déterminant la parité de $a \in R$ à partir de $A = s'(a)$ permet de factoriser N en temps probabiliste polynomial.*

DÉMONSTRATION — Voir [8], [29] ou [45]. □

Chapitre 10

Chiffrement à clé publique

Les cryptosystèmes vus jusqu'ici sont dits *classiques*, ou *à clé secrète*, ou bien encore *symétriques*, car la fonction de déchiffrement se déduit facilement de la fonction de chiffrement. Une personne capable de chiffrer est donc aussi capable de déchiffrer. Avant toute utilisation d'un tel système, le couple (expéditeur, destinataire) doit se mettre d'accord sur une clé, qui ne doit être connue de personne d'autre. Cela ne peut se faire que si les deux personnes disposent d'un canal sûr, ou bien en utilisant un protocole spécifique destiné à générer à distance un tel secret commun.

Le concept de cryptosystème *à clé publique* ou *asymétrique* est apparu pour la première fois dans [41]. Dans un tel système, la fonction de chiffrement peut être largement diffusée. La fonction de déchiffrement est bien sûr tenue secrète par le destinataire. C'est en général lui qui choisit sa fonction de déchiffrement et diffuse à toutes les personnes susceptibles de communiquer avec lui la fonction de chiffrement associée. Son seul souci est alors d'assurer l'authenticité de la clé (si Alice veut envoyer un message à Bob, elle doit s'assurer que la clé qu'elle va utiliser est bien celle de Bob). De plus, lorsque n personnes veulent communiquer ensemble, il est seulement nécessaire d'avoir n couples (clé secrète, clé publique) dans un système asymétrique (un par personne), au lieu de $n(n-1)/2$ clés dans un système classique (une par paire de personnes).

Un tel système ne peut pas être inconditionnellement sûr, puisqu'un attaquant peut en principe décrypter un message c en chiffrant tous les messages clairs possibles jusqu'à ce qu'il obtienne c.

Pour qu'un tel système fonctionne, le destinataire doit connaître une information (la clé secrète) qui doit être très difficile à retrouver par quelqu'un qui ne connaît que la clé publique. La construction d'un tel système peut être basée sur l'existence de fonctions mathématiques dites *à sens unique* car elles sont facilement calculables mais leur réciproque est en pratique impossible à calculer car bien trop coûteuse. Le destinataire choisit alors sa clé secrète et utilise une fonction à sens unique pour calculer la clé publique qu'il diffuse. La construction d'un tel système peut aussi être basée sur l'existence de fonctions *à trappe*, dont la réciproque (et peut-être la fonction elle-même) est difficile à calculer, sauf pour quelqu'un qui connaît une information secrète.

1. Le cryptosystème RSA

Il tire son nom de ses trois inventeurs : R. Rivest, A. Shamir, L. Adleman qui l'ont décrit dans [113]. Le destinataire Bob choisit deux entiers N, E tels que

$$\begin{aligned} &N \text{ est le produit de deux grands nombres premiers distincts } p \text{ et } q \\ &0 \leqslant E \leqslant \varphi(N) \quad \text{et} \quad \operatorname{pgcd}(E, \varphi(N)) = 1, \qquad \text{où } \varphi(N) = (p-1)(q-1). \end{aligned} \tag{1}$$

D'autre part, il calcule l'entier d tel que

$$0 \leqslant d \leqslant \varphi(N) \qquad \text{et} \qquad Ed \equiv 1 \pmod{\varphi(N)}.$$

Il diffuse les entiers N (le *module*) et E (l'*exposant public*), tout en gardant secrets p, q et d (l'*exposant secret*).

1.1. — Définition. Un entier de la forme indiquée par (1) sera appelé un *entier RSA*.

Alice convertit le message qu'elle veut transmettre en éléments de $\mathbb{Z}_N$. Pour chiffrer l'élément $m \in \mathbb{Z}_N$, elle calcule $c = m^E \bmod N$ qu'elle transmet. Bob reçoit c et déchiffre en calculant $c^d \bmod N$. En effet, puisque $\varphi(N)$ est l'ordre du groupe $(\mathbb{Z}/N\mathbb{Z})^*$, on a

$$c^d \equiv m^{Ed} \equiv m \pmod{N}.$$

L'application $E \mapsto d$ est un exemple typique de fonction trappe. Bob peut facilement calculer l'inverse d de E modulo $\varphi(N)$ puisqu'il connaît la factorisation de N mais un attaquant éventuel, ne connaissant que N et E, ne pourra pas calculer d comme le montrent les résultats suivants.

Equivalence des données secrètes

1.2. — Lemme. *La connaissance de $\varphi(N)$ est équivalente à la connaissance de la factorisation de N.*

DÉMONSTRATION — Si la factorisation $N = pq$ est connue alors $\varphi(N) = (p-1)(q-1)$ est facile à calculer. Réciproquement, supposons que $\varphi(N)$ soit connu. Alors, en substituant N/p à q dans la relation $\varphi(N) = (p-1)(q-1)$, on obtient l'équation du second degré

$$p^2 - \big(N + 1 - \varphi(N)\big)p + N = 0$$

qu'il est aisé de résoudre. □

1.3. — Lemme. *Si un attaquant réussit à calculer d connaissant N et E alors il peut factoriser N.*

DÉMONSTRATION — Décomposons $Ed-1$ sous la forme $2^k r$ avec r impair. Puisque $\varphi(N) \mid Ed-1$ on a

$$m^{Ed-1} \equiv 1 \pmod{N} \qquad \text{pour tout } m \text{ tel que } \operatorname{pgcd}(m, N) = 1.$$

Pour un tel m, on a donc trois possibilités :

1. $m^r \equiv 1 \pmod{N}$.
2. Il existe l tel que $0 \leqslant l < k$ et $m^{2^l r} \equiv -1$ modulo N.
3. Il existe l tel que $0 \leqslant l < k$, $m^{2^l r} \not\equiv -1$ et $m^{2^{l+1} r} \equiv 1$ modulo N.

En reprenant la démonstration du théorème 14.1.6, on peut montrer que les deux premiers cas se produisent au plus pour la moitié des valeurs possibles de m. Lorsque le dernier cas se produit, $x = m^{2^l r}$ est une racine carrée de 1 modulo N distincte de ± 1.

Pour factoriser N, l'attaquant peut donc procéder ainsi. Il choisit un m au hasard dans $\mathbb{Z}_N$. Si $\operatorname{pgcd}(m, N) > 1$ alors la factorisation est achevée. Sinon, il calcule $x_0 = m^r \bmod N$ puis la suite $x_{i+1} = x_i^2 \bmod N$ tant que $x_i \not\equiv \pm 1$ modulo N. S'il ne trouve pas d'entier l vérifiant les conditions de la possibilité (3) alors, il recommence en choisissant une nouvelle valeur de m. Mais d'après le paragraphe précédent, cet échec ne se produit qu'au plus une fois sur deux. Sinon, il dispose d'une racine carrée x de 1 modulo N, distincte de ± 1. Les entiers $(x-1)$ et $(x+1)$ sont donc des diviseurs de zéro modulo N et il obtient un facteur non trivial de N en calculant par exemple $\operatorname{pgcd}(x-1, N)$. □

Factorisation et problème RSA

Les données publiques d'une clé RSA sont le module N et l'exposant de chiffrement E. Des données secrètes sont les facteurs p et q, l'exposant de déchiffrement d et l'indicateur d'Euler $\varphi(N)$. Les lemmes ci-dessus montrent que la connaissance de l'une de ces données secrètes permet de retrouver facilement les trois autres. Donc, "casser" une clé RSA (c'est-à-dire retrouver la partie secrète à partir des données publiques) est équivalent à la factorisation du module. Autrement dit, la cryptanalyse totale de RSA est équivalente au problème de la factorisation.

Cependant, personne n'est jamais parvenu à prouver que tout attaquant capable de décrypter des messages chiffrés par RSA est aussi en mesure de factoriser le module. Décrypter des messages chiffrés par RSA (c'est-à-dire réussir une cryptanalyse partielle), c'est extraire des racines E-ièmes modulo N. On nomme ce problème le *problème RSA*.

1.4. — Problème. Soient N un entier RSA et E un entier compris entre 0 et $\varphi(N)$ et relativement premier à $\varphi(N)$. Le *problème RSA* est celui, étant donné un entier c relativement premier à N, de trouver un entier m tel que

$$m^E \equiv c \pmod{N}.$$

En utilisant le théorème chinois on montre facilement que si l'on parvient à factoriser un module RSA, on peut résoudre le problème RSA correspondant. Par contre, le problème RSA est peut-être plus facile à résoudre que celui de la factorisation. Des travaux ont suggéré que cela pourrait être effectivement le cas, en particulier pour de très petits exposants E. Mais la question est loin d'être réglée.

Attaques sur le module RSA

Pour que la factorisation de N soit un problème difficile, il faut qu'un certain nombre de conditions soit remplies.

• Les nombres p et q (donc N) doivent être grands. Typiquement, la taille de p et q est de l'ordre de 100 chiffres décimaux chacun. Aujourd'hui, le crible algébrique casse des clés RSA de plus de 512 bits (environ 154 chiffres décimaux). On préconise donc au minimum 768 bits. Une longueur de clé de 1024 bits offre de bonnes garanties de sécurité. Pour des applications particulièrement sensibles que l'on veut protéger à moyen ou long terme, il faut envisager d'utiliser 1536 bits, voire 2048. L'utilisation de clés de taille encore supérieure ne peut probablement pas se justifier sans évoquer la paranoïa, sauf pour une protection à très long terme (plusieurs dizaines d'années).

• Il faut que $p-q$ aussi soit grand pour contrer la méthode de factorisation de Fermat et ses dérivées (voir la section 15.1).

• Il faut que les nombres $p\pm 1$ et $q\pm 1$ aient chacun un grand facteur premier (disons plus de 100 bits), pour contrer la méthode de factorisation de Pollard et ses dérivées. Mais en fait, si les facteurs p, q sont grands et pris au hasard, les méthodes $p\pm 1$ ainsi que ECM (Lenstra) sont sans espoir de réussite en pratique.

• Soit r le grand facteur premier de $p-1$ (resp. de $q-1$) indiqué ci-dessus. Il faut que $r-1$ ait aussi un grand facteur premier. Sinon, on risque trouver un entier u tel que $E^u \equiv 1$ modulo r (voir la méthode $p-1$ de factorisation de Pollard). En forçant un peu le hasard, on risque même de trouver u tel que $p-1 \mid E^u-1$. Mais alors, $m^{E^u-1} \equiv 1$ modulo p et $p \mid \mathrm{pgcd}(m^{E^u}-m, N)$. On peut donc factoriser N en itérant la fonction de chiffrement (attaque cyclique).

• Il faut que les ordres de E modulo $p-1$ et $q-1$ soient grands. Sinon, l'attaque cyclique pourrait fonctionner là-aussi.

• D'autres conditions rares peuvent fragiliser un module RSA. Elle présentent le danger suivant. Quelqu'un peut sciemment choisir un tel module puis faire semblant de découvrir sa faiblesse plus tard, et se plaindre alors d'avoir été espionné. Nous verrons que l'on peut aussi signer des messages en utilisant RSA. Dans ce cas, les faiblesses rares peuvent permettre au signataire de renier sa signature.

Attaques sur les exposants RSA

Il peut sembler bien imprudent d'utiliser un exposant secret petit. Le résultat suivant en est une confirmation concrète [131].

1.5. — Théorème (Wiener). *Soit $N = pq$ un module RSA tel que $p < q < 2p$. Si l'exposant secret d vérifie $d \leqslant \frac{1}{3}N^{1/4}$ alors on peut calculer d à partir des données publiques.*

Démonstration — Soit k l'entier vérifiant $Ed-1 = k\varphi(N)$. On a $N-\varphi(N) = p+q-1 < 3p < 3\sqrt{N}$, donc

$$\left|\frac{E}{N}-\frac{k}{d}\right| = \frac{|Ed-kN|}{Nd} = \frac{|1-k(N-\varphi(N))|}{Nd} \leqslant \frac{3k\sqrt{N}}{Nd} = \frac{3k}{\sqrt{N}d}.$$

De plus, $k\varphi(N) = Ed-1 < Ed < \varphi(N)d$, d'où $k < d < \frac{1}{3}N^{1/4}$. Ainsi,

$$\left|\frac{E}{N}-\frac{k}{d}\right| \leqslant \frac{1}{N^{1/4}d} < \frac{1}{3d^2}.$$

D'après le théorème 6.4.7, il est possible de calculer k et d. □

1.6. — Exercice. L'exposant de chiffrement $E = 3$ est couramment utilisé dans le système RSA pour minimiser le coût du chiffrement. Supposons qu'Alice transmette le même message à trois destinataires différents qui ont chacun leur module public N_1, N_2 et N_3 mais le même exposant de chiffrement $E = 3$. En utilisant le théorème Chinois, montrer qu'une personne en possession des trois versions chiffrées du message, a toutes les chances de pouvoir le décrypter.

Divers

1.7. — Exercice. Notons $\mathcal{E}(m) = m^E \bmod N$ le message chiffré correspondant à m. Montrer que, pour chaque couple (m_1, m_2) de messages clairs on a

$$\mathcal{E}(m_1 m_2) = \mathcal{E}(m_1)\mathcal{E}(m_2).$$

En déduire une attaque partielle à messages chiffrés choisis.

1.8. — Exercice. Un message est dit *non dissimulé* lorsque $\mathcal{E}(m) = m$. Montrer que le nombre de messages non dissimulés est

$$\big(1 + \mathrm{pgcd}(E-1, p-1)\big)\big(1 + \mathrm{pgcd}(E-1, q-1)\big).$$

Donner des valeurs de E pour lesquelles ce nombre est minimal.

Si l'attaquant possède une information partielle sur le message clair, ne laissant que peu de messages clairs possibles, il peut envisager de chiffrer chacun de ceux-ci afin de découvrir celui qui à été transmis. On peut prévenir cette attaque en "salant" le message (lui adjoindre une composante aléatoire).

Une implémentation rapide de RSA, utilisant un module de 512 bits permet de chiffrer environ 600 K-bits par seconde. C'est près de 1500 fois plus lent qu'une implémentation rapide de DES, qui atteint 1 G-octet par seconde, pour une (in)sécurité semblable. En conséquence, le système RSA est la plupart du temps utilisé pour chiffrer des messages courts. Ce message court peut être lui-même une clé d'un système classique, que l'on utilise ensuite pour chiffrer un message beaucoup plus long.

Le coût de l'exponentiation modulaire est en principe une fonction cubique de la taille du module. Le destinataire qui connaît p et q à donc intérêt à déchiffrer en calculant modulo p et modulo q séparément et à en déduire le résultat modulo N plutôt que de faire le calcul directement modulo N.

Information partielle sur le texte clair

Il semble difficile de retrouver un message clair m à partir du message chiffré $c = m^E \bmod N$ par RSA sans connaître la clé secrète. Mais peut-être est-il facile de retrouver une information partielle sur le message clair, comme par exemple la parité de m, ou bien la position de m par rapport à $N/2$? On va montrer que, pour ces deux cas particuliers, cela n'est pas essentiellement plus facile que de retrouver un message clair complet.

Soient donc les deux fonctions suivantes

$$\mathrm{parité}(c) = \begin{cases} 0 & \text{si } m \text{ est pair,} \\ 1 & \text{si } m \text{ est impair,} \end{cases} \qquad \mathrm{position}(c) = \begin{cases} 0 & \text{si } m < N/2, \\ 1 & \text{si } m > N/2. \end{cases}$$

1.9. — Théorème. *On fixe une instance du cryptosystème RSA (i.e. les entiers N et E). Si on dispose d'un algorithme efficace pour calculer l'une des fonctions parité ou position alors on dispose d'un algorithme efficace pour décrypter n'importe quel message.*

Démonstration — Puisque N est impair, on voit facilement que l'on a les relations

$$\begin{aligned} \mathrm{parité}(2^E c \bmod N) &= \mathrm{position}(c), \\ \mathrm{position}(2^{-E} c \bmod N) &= \mathrm{parité}(c). \end{aligned}$$

Il suffit donc de démontrer le résultat pour la fonction *position*. Or il est clair que la fonction *position* nous donne les informations suivantes :

$$\begin{aligned} \mathrm{position}(c) &= 0 \text{ ssi } m/N \in [0, 1/2[, \\ \mathrm{position}(2^E c \bmod N) &= 0 \text{ ssi } m/n \in [0, 1/4[\cup [1/2, 3/4[, \\ \mathrm{position}(4^E c \bmod N) &= 0 \text{ ssi } m/n \in [0, 1/8[\cup [1/4, 3/8[\cup [1/2, 5/8[\cup [3/4, 7/8[, \\ &\text{etc.} \end{aligned}$$

En guise de démonstration, nous donnons l'algorithme suivant qui permet de décrypter à l'aide de la fonction *position*.

Entrée : le message chiffré c.
Sortie : le message clair $m = c^d \bmod N$.

$k \leftarrow \lfloor \log_2(N) \rfloor + 1$ (le nombre de bits de N)
bas $\leftarrow 0$
haut $\leftarrow 2^k$
Répéter k fois
  $--$ Ici, $m \in [\text{bas} \cdot N/2^k, \text{haut} \cdot N/2^k[$.
  milieu $\leftarrow$ (bas + haut)/2
  si $position(c) = 1$ alors bas $\leftarrow$ milieu sinon haut $\leftarrow$ milieu
  $c \leftarrow 2^E c \bmod N$
Fin Répéter
Retourner $\lceil \text{bas} \cdot N/2^k \rceil$

□

2. Le cryptosystème de Rabin

La cryptanalyse totale d'une instance du cryptosystème RSA revient à factoriser N. Par contre, personne n'a prouvé que cette factorisation est nécessaire pour une cryptanalyse partielle (déchiffrer un message). La sécurité du système suivant, contre une attaque à messages clairs choisis, est en revanche équivalente à la factorisation de son module. En gros, ce système consiste à remplacer l'exposant de chiffrement de RSA par 2. Toutefois, puisque $\text{pgcd}(2, (p-1)(q-1))$ est toujours strictement supérieur à 1, l'élévation au carré modulo N n'est plus une fonction injective.

Le destinataire Bob choisit un entier N qui soit le produit de deux grands nombres premiers distincts, congrus à 3 modulo 4. Il choisit aussi un entier b dans l'intervalle $[0, N-1]$ (en fait, $b = 0$ est souvent un bon choix). Il diffuse les entiers N et b, tout en gardant secrets p et q. Il lui sera utile de calculer les coefficients de Bézout u, v tels que $up + vq = 1$.

Pour chiffrer l'élément $m \in \mathbb{Z}_N$, Alice calcule $c = m(m+b) \bmod N$. Lorsque Bob reçoit c, il déchiffre en calculant les quatre racines carrées modulo N de $c + b^2/4$ (ici, les divisions par 2 et par 4 s'entendent modulo N). Pour cela, il peut calculer d'abord les racines r_p et r_q modulo p et q, ce qui est particulièrement facile puisque $p \equiv q \equiv 3$ modulo 4.

$$r_p = c^{(p+1)/4} \bmod p, \qquad r_q = c^{(p+1)/4} \bmod q.$$

Puis, il utilise le théorème chinois pour obtenir les quatre racines modulo N :

$$r = \pm upr_q \pm vqr_p \bmod N.$$

Le message initial est l'une des quatre valeurs $r - b/2$. En effet, on a

$$(r - b/2)(r - b/2 + b) \equiv (r - b/2)(r + b/2) \equiv r^2 - b^2/4 \equiv (c + b^2/4) - b^2/4 \equiv c \pmod N.$$

Décrypter suivant le schéma de Rabin revient à calculer une racine carrée de $c - b^2/4$ modulo N. Quelqu'un qui saurait comment décrypter saurait donc calculer des racines carrées modulo N. D'après la remarque 8.5.2, il pourrait alors facilement factoriser N.

Un inconvénient évident du schéma de Rabin est que le déchiffrement est ambigu. Précisément, pour $m \in \mathbb{Z}_N$, les quatre messages

$$m, \quad -m - b, \quad \omega m + (\omega - 1)b/2, \quad -\omega m - (\omega + 1)b/2$$

donnent le même message chiffré, ω étant une racine carrée non triviale de 1 modulo N. Au déchiffrement, il faut pouvoir lever l'ambiguïté sur les quatre messages clairs possibles. Cela peut se faire en pratique en ajoutant de la redondance aux messages clairs (par exemple redoubler les 64 derniers bits). Avec une très

grande probabilité, une seule des quatre solutions possibles au déchiffrement respectera cette redondance. Une autre méthode pour lever l'ambiguïté consiste à transmettre, en plus du message chiffré, deux bits supplémentaires permettant de caractériser le message réel m parmi les quatres solutions. La parité de $m + b/2$ et son symbole de Jacobi vis-à-vis de N sont deux bits permettant une telle caractérisation.

Si le schéma de Rabin est sûr (si on admet que le problème de la factorisation est difficile) vis-à-vis d'une attaque à messages clairs choisis, il est par contre vulnérable contre une attaque à messages chiffrés choisis. En effet, si un attaquant choisit un message m au hasard, calcule $c = m(m + b)$ et obtient un déchiffrement m' de c, alors il y a une chance sur deux pour que m' soit différent de m et de $-m - b$. Il peut dans ce cas calculer ω et donc factoriser N. L'ajout de redondance rend cette attaque inutilisable. Toutefois, on ne sait pas rigoureusement prouver que la sécurité du schéma de Rabin avec redondance est assujetti à la factorisation.

3. Le cryptosystème El Gamal

La sécurité de RSA est liée à la factorisation. Celle de El Gamal est liée à un autre problème réputé difficile de théorie des nombres, celui du logarithme discret.

Le problème du logarithme discret

3.1. — Problème. Soient p un (grand) nombre premier et g une racine primitive modulo p. Le *problème du logarithme discret* est celui de trouver, étant donné $A \in \mathbb{Z}$ tel que $p \nmid A$, l'entier a vérifiant

$$g^a \equiv A \pmod{p} \qquad \text{avec } 0 \leqslant a \leqslant p-2.$$

Bien que ce problème ne ressemble pas à priori au problème de la factorisation, il se trouve que sa difficulté semble très comparable. Il est à noter que tout progrès réalisé dans la résolution de l'un de ces problèmes à été suivi rapidement d'un progrès dans la résolution de l'autre. En fait, il est même assez troublant de constater à quel point les algorithmes respectifs dont on dispose pour attaquer ces deux problèmes ont beaucoup de points communs, que ce soit dans leurs principes ou dans leurs coûts (voir les deux chapitres consacrés à ces sujets).

3.2. — Problème. Soient p un (grand) nombre premier et g une racine primitive modulo p. Le *problème de Diffie-Hellman* est celui de trouver, étant donnés $A, B \in \mathbb{Z}$ non divisibles par p, l'entier C vérifiant

$$C = g^{ab} \bmod p \qquad \text{où } a, b \in \mathbb{Z} \text{ (inconnus) vérifient } g^a \equiv A \text{ et } g^b \equiv B \text{ modulo } p.$$

Si on sait résoudre le problème du logarithme discret, on sait aussi résoudre le problème de Diffie-Hellman. La réciproque semble vraie, comme cela sera argumenté dans le chapitre suivant.

On peut aussi formuler ces deux problèmes de manière plus générale dans un corps fini quelconque.

Le cryptosystème El Gamal

Le destinataire Bob choisit un (grand) nombre premier p ainsi qu'une racine primitive g modulo p. Il choisit aussi un entier b dans $[1, p-2]$ et calcule $B = g^b \bmod p$. Il publie les entiers p, g et B mais garde b secret.

Pour chiffrer un message $m \in \mathbb{Z}_p$, Alice choisit un entier k dans $[1, p-2]$ et calcule $K = g^k \bmod p$. Elle calcule aussi $c = mB^k \bmod p$ et envoie le couple (K, c) à Bob.

Bob peut alors retrouver m en calculant $cK^{-b} \bmod m$. En effet,

$$cK^{-b} \equiv cg^{-bk} \equiv cB^{-k} \equiv m \pmod{p}.$$

Il est important que k soit choisi au hasard pour chaque élément de $\mathbb{Z}_m$ à chiffrer. Si le même k est utilisé pour chiffrer m_1 et m_2 alors les messages chiffrés c_1 et c_2 vérifient $c_1 c_2^{-1} \equiv m_1 m_2^{-1}$ modulo p. Donc toute information connue concernant m_1 peut être utilisée pour décrypter m_2 et réciproquement.

Retrouver la clé secrète à partir de la clé publique (cryptanalyse totale) revient clairement à résoudre une instance du problème du logarithme discret. Un attaquant éventuel qui intercepterait un message chiffré disposerait de B, K et c. Puisque $c/m \equiv g^{bk}$ modulo p, décrypter c (cryptanalyse partielle) est équivalent à résoudre une instance du problème de Diffie-Hellman.

Pour assurer la sécurité de ce cryptosystème, le nombre premier p doit être suffisamment grand. La taille requise est analogue à celle d'un module RSA. Avec un nombre premier de 512 bits, la sécurité n'est plus assurée face aux moyens de calculs disponibles aujourd'hui, sauf pour des clés dont la durée de vie est très courte (par exemple une heure). On préconise désormais 768 ou 1024 bits. Pour des applications particulièrement sensibles que l'on veut protéger à long terme, il faut envisager d'utiliser 2048 bits. D'autre part, pour contrer la méthode de Pohlig-Hellman de calcul du logarithme discret, il faut que $p-1$ ait un grand facteur premier.

Information partielle sur le logarithme discret

Soit p un nombre premier et g une racine primitive modulo p. Calculer le logarithme discret a d'un entier A semble être un problème difficile. On peut se demander toutefois si on peut obtenir facilement une information partielle sur a. La réponse est *oui* pour la parité de a puisque a est pair si et seulement si A est un carré modulo p. Un simple calcul de symbole de Jacobi permet donc de déterminer la parité de a. Plus généralement, on peut déterminer facilement le reste modulo q de a lorsque q est un petit diviseur (ou un produit de petits diviseurs) de $p-1$. C'est la base de l'algorithme de Pohlig-Hellman du calcul du logarithme discret. En particulier, si 2^k divise $p-1$, on peut déterminer facilement les k bits de poids faible de a. Toutefois, on peut montrer que les autres bits sont essentiellement aussi difficiles à obtenir que le logarithme complet. L'algorithme suivant le montre pour le bit de poids 2 lorsque $p \equiv 3$ modulo 4.

3.3. — Algorithme. Calcul du logarithme discret, lorsque $p \equiv 3$ modulo 4, utilisant un oracle donnant le bit de poids 2.

Entrée : Un entier A non divisible par p.
Sortie : Le logarithme discret a de A en base g.

Déterminer le bit de poids faible a_0 de a en calculant le symbole de Legendre (A/p)
$A \leftarrow Ag^{-a_0} \bmod p$
$i \leftarrow 1$
Tant que $A \neq 1$ répéter
  – – [A est un carré modulo p.]
  $a_i \leftarrow$ bit de poids 2 de A
  $A \leftarrow A^{(p+1)/4} \bmod p$ (racine carrée modulo p)
  Déterminer le bit de poids faible du logarithme de A en calculant (A/p)
  S'il est contraire au bit a_i alors faire $A \leftarrow p - A$.
  $A \leftarrow Ag^{-a_i} \bmod p$
  $i \leftarrow i + 1$
Fin tant que
Retourner $a = \sum_{j=0}^{i-1} a_j 2^j$

3.4. — Exercice. Montrer que la fonction *position* permet aussi de calculer le logarithme discret.

3.5. — Remarque. On peut définir une variante du cryptosystème de El Gamal, en choisissant pour g un élément d'ordre q grand premier (divisant $p-1$) ce qui revient à travailler dans le groupe $\left((\mathbb{Z}/p\mathbb{Z})^*\right)^{(p-1)/q}$. Dans ce cas l'attaque ci-dessus révélant une information partielle sur le texte clair ne fonctionne plus.

4. Le cryptosystème XTR

Soient p, q deux (grands) nombres premiers tels que $q \mid p^2 - p + 1$. Le groupe $\mathbb{F}_{p^6}^*$ est cyclique et contient un sous-groupe G d'ordre q. Soit g un générateur de G. Une idée de base de XTR est de représenter les éléments de G par leur trace dans $\mathbb{F}_{p^2}$ et de remplacer ainsi les calculs dans $\mathbb{F}_{p^6}$ par des calculs dans $\mathbb{F}_{p^2}$. Une autre idée est de simplifier au maximum les calculs dans $\mathbb{F}_{p^2}$ à l'aide d'une base normale sympathique sur $\mathbb{F}_p$.

Arithmétique rapide dans $\mathbb{F}_{p^2}$

4.1. — Théorème. *Soit p un nombre premier congru à 2 modulo 3. Le corps $\mathbb{F}_{p^2}$ est de la forme $\mathbb{F}_p(\alpha)$ avec $\alpha^2+\alpha+1=0$. De plus, (α,α^2) est une base normale.*

Démonstration — Le discriminant de X^2+X+1 est -3, et par la loi de réciprocité quadratique, ce n'est pas un carré modulo p. Cela montre la première affirmation et la deuxième est claire puisque $\alpha^3=1$ donc $\alpha^p=\alpha^2$. □

4.2. — Proposition. *Soit p un nombre premier congru à 2 modulo 3. Le coût de différentes opérations dans $\mathbb{F}_{p^2}$ en nombre de multiplications dans $\mathbb{F}_p$ est :*

- *Une multiplication dans $\mathbb{F}_{p^2}$ coûte trois multiplications dans $\mathbb{F}_p$.*
- *Une élévation au carré dans $\mathbb{F}_{p^2}$ coûte deux multiplications dans $\mathbb{F}_p$.*
- *Une élévation à la puissance p dans $\mathbb{F}_{p^2}$ ne coûte aucune opération dans $\mathbb{F}_p$.*
- *Calculer $xz-yz^p$ pour $x,y,z\in\mathbb{F}_{p_2}$ coûte quatre multiplications dans $\mathbb{F}_p$.*

Démonstration — Soient $x=x_1\alpha+x_2\alpha^2$ et $y=y_1\alpha+y_2\alpha^2$ deux éléments de $\mathbb{F}_{p^2}$ et leurs coordonnées dans la base normale (α,α^2). On a $\alpha^3=1=-\alpha-\alpha^2$ et

$$xy=(v-w)\alpha+(u-w)\alpha^2$$
$$\text{avec } u=x_1y_1,\quad v=x_2y_2,\quad w=x_1y_2+x_2y_1=(x_1+x_2)(y_1+y_2)-u-v,$$

d'où le coût de la multiplication. Celui de l'élévation au carré résulte de la formule

$$x^2=(x_2-2x_1)x_2\alpha+(x_1-2x_2)x_1\alpha^2.$$

D'autre part, comme $p\equiv 2$ modulo 3, on a $\alpha^p=\alpha^2$. Donc $x^p=x_2\alpha+x_1\alpha^2$. Enfin, en désignant par (x_1,x_2), (y_1,y_2) et (z_1,z_2) les coordonnées respectives de x,y,z dans la base (α,α^2), on calcule facilement que

$$xz-yz^p=\big(z_1(y_1-x_2-y_2)+z_2(x_2-x_1+y_2)\big)\alpha+\big(z_1(x_1-x_2+y_1)+z_2(y_2-x_1-y_1)\big)\alpha^2,$$

ce qui représente quatre multiplications. □

Le sous-groupe XTR

Rappelons d'abord la décomposition de X^6-1 en produit de polynômes cyclotomiques :

$$X^6-1=(X-1)(X+1)(X^2+X+1)(X^2-X+1).$$

4.3. — Lemme. *Soient p,q deux entiers avec $q>3$ impair et diviseur de p^2-p+1. Alors q ne divise pas $(p^6-1)/(p^2-p+1)$.*

Démonstration — On a

$$\frac{p^6-1}{p^2-p+1}=p^4+p^3-p-1=(p^2+2p+1)(p^2-p+1)-2(p+1).$$

Donc, si q divise à la fois p^2-p+1 et p^4+p^3-p-1, il divise aussi $p+1$. D'autre part, on a

$$p^2-p+1=(p-2)(p+1)+3.$$

Donc, q divise aussi 3. Or $q>3$. □

4.4. — **Définition.** Soit p un nombre premier. Le groupe $\mathbb{F}_{p^6}^*$ est cyclique et son ordre p^6-1 est un multiple p^2-p+1. Il contient donc un unique sous-groupe d'ordre p^2-p+1. On appelle parfois ce sous-groupe le *(super-)groupe XTR d'indice p.*

4.5. — **Théorème.** *Soit p un nombre premier et G le groupe XTR correspondant. Tout élément de G contenu dans un sous-corps propre de $\mathbb{F}_{p^6}$ est d'ordre $\leqslant 3$.*

Démonstration — Soit $h \in G$ d'ordre $q > 3$. Puisque p^2-p+1 est impair, q l'est aussi. D'après le lemme, q ne divise ni p^2-1 ni p^3-1. Donc g n'appartient à aucun des deux sous-corps maximaux $\mathbb{F}_{p^2}$ et $\mathbb{F}_{p^3}$ de $\mathbb{F}_{p^6}$. □

4.6. — **Définition.** Soient p, q deux nombres premiers tels que $q > 3$ et $q \mid p^2-p+1$. D'après le théorème précédent, $\mathbb{F}_{p^6}^*$ contient un unique sous-groupe d'ordre q, et ce sous-groupe n'est contenu dans aucun sous-corps propre de $\mathbb{F}_{p^6}$. On appelle parfois ce sous-groupe le *sous-groupe XTR de paramètres p, q.*

Trace sur $\mathbb{F}_{p^2}$

Désignons par tr l'application trace de $\mathbb{F}_{p^6}$ dans $\mathbb{F}_{p^2}$, donc $\mathrm{tr}(h) = h + h^{p^2} + h^{p^4}$. Dans le cas particulier où h est dans le groupe XTR, on a $\mathrm{tr}(h) = h + h^{p-1} + h^{-p}$ puisque $h^{p^2-p+1} = 1$. On a aussi dans ce cas $\mathrm{tr}(h)^p = h^{-1} + h^{1-p} + h^p = \mathrm{tr}(h^{-1})$ et la norme de h est 1. Le polynôme caractéristique de h est alors

$$(X-h)(X-h^{p^2})(X-h^{p^4}) = X^3 - \mathrm{tr}(h)X^2 + \mathrm{tr}(h)^p X - 1.$$

Soit g un élément d'ordre q dans le groupe XTR. Pour $n \in \mathbb{Z}$, on pose

$$t^n = \mathrm{tr}(g^n) = g^n + g^{n(p-1)} + g^{-np}$$

de telle sorte que le polynôme caractéristique de g^n soit $X^3 - t_n X^2 + t_{-n} X - 1$. On a alors les formules suivantes.

4.7. — **Proposition.** *Pour $u, v \in \mathbb{Z}$, on a l'identité*

$$t_{u+v} = t_u t_v - t_v^p t_{u-v} + t_{u-2v}.$$

Démonstration — Le calcul

$$\begin{aligned}
t_u t_v - t_v^p t_{u-v} &= t_u t_v - t_{-v} t_{u-v} \\
&= (g^u + g^{u(p-1)} + g^{-up})(g^v + g^{v(p-1)} + g^{-vp}) \\
&\quad - (g^{-v} + g^{-v(p-1)} + g^{vp})(g^{u-v} + g^{(u-v)(p-1)} + g^{-(u-v)p}) \\
&= g^{u+v} + g^{(u+v)(p-1)} + g^{-(u+v)p} + g^{u+v(p-1)} + g^{u-vp} + g^{u(p-1)+v} \\
&\quad + g^{u(p-1)-vp} + g^{-up+v} + g^{-up+v(p-1)} - g^{u-2v} - g^{(u-2v)(p-1)} - g^{-(u-2v)p} \\
&\quad - g^{(u-v)p-u} - g^{-v-(u-v)p} - g^{-vp+u} - g^{v-up} - g^{v(p-1)+u} - g^{up-(u-v)} \\
&= g^{u+v} + g^{(u+v)(p-1)} + g^{-(u+v)p} - g^{u-2v} - g^{(u-2v)(p-1)} - g^{-(u-2v)p} \\
&= t_{u+v} - t_{u-2v}
\end{aligned}$$

montre cette identité. □

4.8. — **Proposition.** *Pour $n \in \mathbb{Z}$, on a les formules*

$$\begin{aligned}
t_{-n} &= t_n^p \\
t_{2n} &= t_n^2 - 2t_n^p \\
t_{2n+1} &= t_n t_{n+1} - t_{n+1}^p t_1^p + t_{n+2}^p \\
t_{2n+3} &= t_{n+1} t_{n+2} - t_{n+1}^p t_1 + t_n^p.
\end{aligned}$$

D'autre part, calculer t_{2n}, t_{2n+1} et t_{2n+3} (à partir de t_n, t_{n+1}, t_{n+2}), par ces formules coûte respectivement 2, 4 et 4 multiplications dans $\mathbb{F}_p$.

Démonstration — La première formule résulte de l'identité déjà vue $\mathrm{tr}(h)^p = \mathrm{tr}(h^{-1})$. Les trois autres résultent de la proposition 4.7 en posant $u = v = n$ pour la deuxième, $(u, v) = (n, n+1)$ pour la troisième et $(u, v) = (n+2, n+1)$ pour la quatrième. Les coûts indiqués résultent de la proposition 4.2. □

Pour $k \in \mathbb{Z}$, posons $S_k = (t_{2k}, t_{2k+1}, t_{2k+2}) \in \mathbb{F}_{p^2}^3$. D'après la proposition 4.8, on peut calculer au choix S_{2k} ou S_{2k+1} (à partir de S_k), avec 8 multiplications dans $\mathbb{F}_p$. Cela permet de calculer t_n pour $n \in \mathbb{N}$ avec environ $8\log_2(n)$ multiplications dans $\mathbb{F}_p$.

Il reste à remarquer que les schémas classiques utilisant le logarithme discret peuvent s'adapter à ce cadre. Plus précisément, au lieu d'utiliser $n \mapsto g^n$ comme fonction à sens unique, on utilise $n \mapsto t_n$. Le protocole d'échange de clés de Diffie-Hellman s'adapte particulièrement facilement en version XTR. Le protocole de chiffrement El Gamal s'adapte sous la forme suivante.

Version XTR de El Gamal

• Génération de clé. On fixe les paramètres publics : un grand nombre premier $p \equiv 2$ modulo 3 (disons 1024 bits), un diviseur premier q de $p^2 - p + 1$ (disons de taille 160 bits), et un entier g d'ordre q modulo p (donc, un générateur du sous-groupe XTR de paramètres p, q). Le destinataire Bob choisit sa clé secrète $b \in [0, q-1]$ et calcule sa clé publique t_b.

• Chiffrement. Alice choisit $k \in [0, q-1]$ au hasard et calcule t_k. Ensuite, connaissant la valeur de $t_b = \mathrm{tr}(B)$ où $B = g^b \bmod p$, elle peut calculer les $\mathrm{tr}(B^n)$ pour les n de son choix, et en particulier calculer $R = \mathrm{tr}(B^k)$. Elle utilise alors un cryptosystème symétrique et une clé dérivée de R pour chiffrer le message. Elle envoie ce message chiffré accompagné de t_k.

• Déchiffrement. Connaissant $t_k = \mathrm{tr}(K)$ où $K = g^k \bmod p$, Bob peut lui aussi calculer $R = \mathrm{tr}(K^b)$ et donc la clé symétrique utilisée par Alice. Il n'a plus qu'à utiliser cette clé pour retrouver le message clair.

5. Cryptosystèmes basés sur les codes correcteurs

Le cryptosystème de Mc Eliece

Ce cryptosystème à clé publique a été introduit dans [74] et son principe est le suivant. On choisit un code correcteur sur un corps K pour lequel on connaît un algorithme efficace de décodage (par exemple un code de Goppa). Ce code C est donné par une matrice génératrice G de taille $k \times n$, où n et k sont la longueur et la dimension de C. On choisit aussi un automorphisme de K^k, donné par une matrice inversible U, et une isométrie (pour la distance de Hamming) de K^n, donnée par une matrice de permutation P, qui vont servir à masquer la nature du code initial C qui lui, restera secret. La clé publique sera constituée de la matrice $\hat{G} = UGP$, et la clé privée des trois matrices U, G, P.

L'espace des messages clairs est K^k, et chaque bloc m sera chiffré en l'encodant avec la matrice $\hat{G}$ et en ajoutant une "erreur" aléatoire e de poids t, la capacité correctrice de C :

$$c = m\hat{G} + e \qquad \text{avec } w(e) = t.$$

Le message chiffré c obtenu est un élément de K^n (en fait, du code $\hat{C}$, de matrice génératrice UGP). Pour le déchiffrer, le destinataire lui applique la permutation définie par P^{-1} :

$$cP^{-1} = mUG + eP^{-1}$$

et décode le mot obtenu. Il obtient mU puisque eP^{-1} est de poids t. Il ne lui reste plus qu'à appliquer la transformation définie par U^{-1} pour retrouver m.

Il est important que l'expéditeur du message respecte le protocole de chiffrement, en ajoutant un mot aléatoire e de poids t après encodage. S'il ne le fait pas, son message clair m peut être retrouvé à partir du message chiffré, sans toutefois que la clé privée soit compromise. Il suffit pour cela qu'un attaquant sélectionne k colonnes de la matrice $\hat{G}$ formant une matrice carrée $\hat{G}_k$ inversible. Le seul message clair m' tel que l'encodé correspondant $m'\hat{G}$ coïncide avec c sur les k positions sélectionnées est m et il est facile de le retrouver en résolvant $m\hat{G}_k = c_k$, où c_k désigne les k symboles de c correspondant aux colonnes sélectionnées. Si au contraire, l'expéditeur ajoute un mot aléatoire e après chaque encodage, la méthode de décryptage ci-dessus ne fonctionne que si les k positions sélectionnées sont en dehors du support de e. Ceci est hautement improbable lorsque les paramètres sont choisis convenablement.

Les paramètres suggérés à l'origine par Mc Eliece étaient $n = 1024$, $t = 50$ et $k \geqslant 524$. Lorsqu'on base ce cryptosystème sur un code de Goppa, un choix optimal pour la sécurité est $n = 1024$, $t = 38$ et $k \geqslant 644$. Un inconvénient de ce cryptosystème est la très grande taille de la clé publique (environ 2^{19} bits dans le cas des paramètres précédents). Un autre inconvénient est un taux de transmission k/n nettement inférieur à 1. Ces inconvénients ont empêché ce cryptosystème d'être utilisé en pratique. D'autre part, l'étude profonde de sa sécurité n'a encore bénéficié que de peu d'efforts comparativement aux cryptosystèmes usuels.

Cryptosystème de Niederreiter

Ce cryptosystème à été introduit dans [96] et sa sécurité est équivalente à celle du précédent. Toutefois, avec des choix de paramètres judicieux, la taille des clés est un peu plus raisonnable que dans le cas de Mc Eliece. Le code C peut être identique à celui utilisé pour le cryptosystème de Mc Eliece mais il est ici donné par une matrice de contrôle H (de taille $(n-k) \times n$). On masque la matrice H en utilisant une matrice inversible V de taille $(n-k) \times (n-k)$ et une matrice de permutation P de taille $n \times n$ (en fait, la même que ci-dessus). La matrice $\hat{H} = VHP$ obtenue est une matrice de contrôle du code $\hat{C}$. Elle est rendue publique et H est gardée secrète.

L'espace des messages clairs est cette fois-ci l'ensemble de mots de longueur n et de poids t (donc des "erreurs"). Le chiffré correspondant au message m est son syndrôme relativement à la matrice publique :

$$c = \hat{H}m^T \qquad \text{avec } w(m) = t.$$

Pour déchiffrer, on remarque que $V^{-1}c$ est le syndrôme de Pm^T relativement à la matrice secrète :

$$V^{-1}c = HPm^T.$$

L'algorithme de décodage dans C (il faut que C possède un algorithme de décodage par syndrôme efficace) permet de retrouver l'"erreur" Pm^T de poids t. Il ne reste plus qu'à appliquer P^{-1} pour retrouver m.

6. Cryptosystèmes utilisant les courbes elliptiques

Nous verrons que la résolution du logarithme discret dans les corps finis est accessible sur des corps de taille moyenne grâce à des algorithmes relativement récents (algorithmes sous-exponentiels). L'existence de ces algorithmes oblige à utiliser des modules assez grands dans les cryptosystèmes RSA et El Gamal pour préserver leur sécurité. Mais les problèmes du logarithme discret et de Diffie-Hellman ont des homologues naturels en termes de courbes elliptiques. Par exemple :

6.1. — Problème. Soient E une courbe elliptique sur un corps fini et G un point d'ordre h (grand) sur E. Le *problème du logarithme discret sur E* est celui de trouver, étant donné un point A du sous-groupe de E engendré par G, l'entier a vérifiant

$$aG = A \qquad \text{avec } 0 \leqslant a \leqslant h-1.$$

Or, on ne dispose pas actuellement de méthode sous-exponentielle pour résoudre le logarithme discret sur les courbes elliptiques. Le problème du logarithme discret sur les courbes elliptiques est donc (provisoirement ?) plus difficile, à taille égale, que le problème du logarithme discret classique. Il est donc tentant de transcrire les cryptosystèmes basés sur le logarithme discret en termes de courbes elliptiques, pour obtenir des cryptosystèmes de sécurité équivalente avec des clés plus petites.

Une transcription directe du cryptosystème de El Gamal en termes de courbes elliptiques serait imaginable mais pose quelques problèmes techniques (grande taille du message chiffré par rapport à celle du message clair, génération de points de E en fonction du message à transmettre, ...). On lui préfère donc la méthode suivante.

Cryptosystème de Menezes/Vanstone

Soient E une courbe elliptique sur un corps fini et G un point d'ordre h (grand) sur E, rendus publics. Le destinataire Bob choisit un entier secret $b \in [0, h-1]$ et diffuse la valeur du point $B = bG$. L'expéditeur Alice peut alors chiffrer un message $M = (x_M, y_M) \in \mathbb{F}_q \times \mathbb{F}_q$ de la manière suivante (noter que le point M n'est pas nécessairement sur E). Elle choisit secrètement un entier $k \in [0, h-1]$, calcule $K = kG$ et $kB = (x_{kB}, y_{kB})$. Elle transmet à Bob les données (K, C) où $C = (x_C, y_C) = (x_M x_{kB}, y_M y_{kB})$. A la réception, Bob peut retrouver M en calculant $bK = (x_{bK}, y_{bK})$ et $(x_M, y_M) = (x_C x_{bK}^{-1}, y_C y_{bK}^{-1})$.

7. Echanges de clés

Le protocole de Diffie-Hellman est basé sur le problème 3.2. Il permet à deux personnes éloignées de générer un secret commun. Ce secret peut ensuite être par exemple utilisé comme clé par ces deux personnes pour communiquer à l'aide d'un système cryptographique classique.

Diffie-Hellman classique

Le protocole est le suivant. Les deux personnes Alice et Bob se mettent d'accord sur un grand nombre premier p et sur une racine primitive modulo p (qu'elles peuvent se transmettre sur le réseau public). Ensuite elles choisissent chacune secrètement un entier dans l'intervalle $[1, p-2]$. Soient a, b les entiers qu'elles ont respectivement choisis. Alice calcule $A = g^a \bmod p$ transmet A à Bob. De même, Bob calcule $B = g^b \bmod p$ et transmet B à Alice. Lorsque Alice reçoit B, elle calcule $C = B^a \bmod p$. De même, lorsqu'il reçoit A, Bob calcule la même valeur $C = A^b \bmod p$. Le secret commun est alors C et un éventuel espion qui aurait capté les valeurs de A et B serait confronté à une instance du problème de Diffie-Hellman s'il voulait déterminer C.

Paramètres publics : p grand premier, g entier d'ordre $p-1$ modulo p		
Alice		**Bob**
$a \in [1, p-2]$		$b \in [1, p-2]$
$A = g^a \bmod p$	$\xrightarrow{A}$	
	$\xleftarrow{B}$	$B = g^b \bmod p$
$C = B^a \bmod p$		$C = A^b \bmod p$

Schéma 1. Protocole de Diffie-Hellman

Version elliptique

On peut aussi transcrire ce protocole en termes de courbes elliptiques en le basant sur l'homologue elliptique du problème de Diffie-Hellman :

7.1. — Problème. Soient E une courbe elliptique sur un corps fini et G un point d'ordre r (grand) sur E. Le *problème de Diffie-Hellman sur* E est celui de trouver, étant donnés $A, B \in \langle G \rangle$, le point $C \in E$ vérifiant

$$C = abG \qquad \text{où } a, b \in \mathbb{Z} \text{ (inconnus) vérifient } aG = A \text{ et } bG = B.$$

Alice et Bob peuvent générer comme secret commun un point d'une courbe elliptique en s'appuyant sur le schéma suivant. Dans la pratique, le secret réellement utilisé pourra être l'abscisse du point obtenu.

Paramètres publics : Courbe elliptique E sur un corps $\mathbb{F}_q$, G point de E d'ordre r grand		
Alice		**Bob**
$a \in [1, r-1]$		$b \in [1, r-1]$
$A = aG$	$\xrightarrow{A}$	
	$\xleftarrow{B}$	$B = bG$
$C = aB$		$C = bA$

Schéma 2. Protocole Diffie-Hellman elliptique

7.2. — **Exercice.** On suppose que le groupe de la courbe elliptique E est produit direct $\langle G\rangle \times \langle H\rangle$ du sous-groupe engendré par G et d'un autre sous-groupe, d'ordre k petit et relativement premier à r, engendré par H.

• Montrer que si Bob est malicieux, il peut obtenir la valeur de $a \bmod k$ en substituant $B' = B + H$ à B dans le protocole précédent et en forçant Alice à communiquer en utilisant la valeur qu'elle obtient.

• Montrer que Bob ne peut plus mener cette attaque si on remplace les calculs du secret C par $C = kaB$ pour Alice et $C = kbA$ pour Bob.

Version XTR

Enfin, voici la version XTR du protocole de Diffie-Hellman.

Paramètres publics :		
$p \equiv 2$ modulo 3, grand nombre premier (e.g. 1024 bits)		
$q \mid p^2 - p + 1$ grand nombre premier (e.g. 160 bits)		
g entier d'ordre q modulo p (ou seulement la trace de g)		
Alice		**Bob**
$a \in [0, q-1]$		$b \in [0, q-1]$
$A = g^a \bmod p$		$B = g^b \bmod p$
$t_a = \mathrm{tr}(A)$	$\xrightarrow{t_a}$	
	$\xleftarrow{t_b}$	$t_b = \mathrm{tr}(B)$
$C = \mathrm{tr}(B^a)$		$C = \mathrm{tr}(A^b)$

Schéma 3. Protocole XTR Diffie-Hellman

Chapitre 11

Compléments concernant la sécurité

Ce chapitre développe certains points concernant la sécurité des protocoles de chiffrement asymétriques.

1. Sécurité sémantique

Nous avons vu que le problème de découvrir les bits de poids faible d'un message chiffré sous RSA est aussi difficile que celui de découvrir un message complet. Mais on ne sait pas s'il en est de même de toute autre information partielle. D'ailleurs, nous avons vu au contraire que le bit de poids faible d'un message chiffré sous El Gamal est très facile à découvrir. On peut se demander s'il existe des cryptosystèmes qui puissent assurer la sécurité de n'importe quelle information partielle sur le texte clair. Cette propriété à été introduite dans [48] et est appelée la sécurité sémantique.

1.1. — Définition. On dit qu'un cryptosystème est *sémantiquement sûr* si toute information sur le texte clair pouvant être extraite (en temps moyen polynomial) du message chiffré correspondant peut aussi l'être sans la connaissance du message chiffré.

Une conséquence de cette propriété est la suivante. Si m_1 et m_2 sont deux messages clairs et que c est le message chiffré correspondant à l'un d'eux, un attaquant est incapable (en temps polynomial) de préciser si c'est à m_1 ou bien à m_2 que c correspond. En particulier, un cryptosystème sémantiquement sûr est non-déterministe.

Le cryptosystème de Goldwasser-Micali

Soit $N = pq$ un entier RSA (un produit de deux grands nombres premiers distincts de taille voisine) et définissons $\mathbb{Z}_N^+$, Q, et $\overline{Q}$ comme dans 5.3.19 :

$$\mathbb{Z}_N^+ = \left\{ x \in \mathbb{Z}_N \,\middle|\, \left(\frac{x}{N}\right) = 1 \right\}, \qquad Q = \{x \in \mathbb{Z}_N \mid x = y^2 \bmod N \text{ avec } y \in \mathbb{Z}_N^*\}, \qquad \overline{Q} = \mathbb{Z}_N^+ \setminus Q.$$

L'entier N est rendu public et les facteurs p, q sont gardés secrets par le destinataire des messages. Celui-ci détermine aussi un élément g de $\overline{Q}$ et le rend public.

Pour chaque bit b_i du message clair ($0 \leqslant i \leqslant t$), l'expéditeur choisit :

- Un élément x_i aléatoire de Q si $b = 1$ (en choisissant $y_i \in \mathbb{Z}_N$ au hasard et en posant $x_i = y_i^2 \bmod N$).
- Un élément x_i aléatoire de $\overline{Q}$ si $b = 0$ (en choisissant $y_i \in \mathbb{Z}_N$ au hasard et en posant $x_i = g y_i^2 \bmod N$).

Le message chiffré transmit est le t-uplet $(x_1, \ldots, x_t)$. En résumé chaque bit du message clair est chiffré par un élément x de $\mathbb{Z}_N$. Un bit 1 est représenté par un carré aléatoire (un élément de Q) ; un bit 0 par un non-carré tel que $(x/N) = 1$ (un élément de $\mathbb{Z}_N^+ \setminus Q$).

Le cryptosystème de Goldwasser-Micali est sémantiquement sûr si le problème 5.3.19 de la résidualité quadratique est difficile (ou si la factorisation de N est difficile, au prix d'une modification analogue à celle du théorème 9.2.7). En effet, supposons que m_0 et m_1 sont deux messages clairs distincts connus d'un attaquant A de la sécurité sémantique et que le chiffré c de l'un d'eux m_β est donné à A. Alors A peut choisir un indice i tel que les bits de rang i dans m_0 et m_1 soient distincts. Le problème de déterminer β est alors équivalent à celui de déterminer si le composant de c de rang i est un élément de Q ou bien de $\overline{Q}$.

Ce cryptosystème est très simple mais le message chiffré est beaucoup plus long que le message clair ce qui le rend sans intérêt pratique. Il existe d'autres cryptosystèmes sémantiquement sûrs pour lesquels l'expansion du message lors du chiffrement est faible. Parmi eux, celui de Blum-Goldwasser dont la sécurité sémantique repose aussi sur le problème de la résidualité quadratique.

Blum-Goldwasser

Soit N un entier de Blum public dont la factorisation est connue seulement du destinataire. Le message à chiffrer est une suite de ℓ bits.

• Chiffrement. On choisit un germe aléatoire $a_0 \in \mathbb{Z}_N^+$. On itère BBS pour obtenir des bits pseudo-aléatoires $r_1, \ldots, r_\ell$, et on prolonge le calcul pour obtenir $a_{\ell+1} = a_\ell^2$ modulo N. On fait du chiffrement à flot (OU EXCLUSIF bit-à-bit) entre le clair et les bits pseudo-aléatoires. On transmet les bits obtenus suivis de $a_{\ell+1}$.

• Déchiffrement. Le destinataire calcule $e_p = \big((p+1)/4\big)^\ell \bmod p-1$ puis $a_{\ell+1}^{e_p} \bmod p \equiv a_1$ modulo p (voir remarque 8.5.1). Il fait de même pour q et en déduit a_1. Il itère alors BBS pour retrouver les bits du message clair.

En reprenant la démonstration du théorème 9.2.5, on peut montrer que le générateur BBS reste sûr si la valeur $a_{\ell+1}$ est connue de l'attaquant. On peut en déduire que ce cryptosystème est sémantiquement sûr sous l'hypothèse que le problème de la résidualité quadratique modulo N est difficile. Là encore, en remplaçant le générateur BBS par sa variante, le théorème 9.2.7 s'applique et on obtient un cryptosystème sémantiquement sûr sous l'hypothèse que N est difficile à factoriser.

Le cas de El Gamal

1.2. — **Problème (DDH).** Soit G un groupe cyclique (noté multiplicativement) de générateur g et d'ordre q. Le problème décisionnel de Diffie-Hellman est celui, étant donné un triplet

$$(A, B, C) \in G^3 \qquad \text{et en posant } a, b, c \in \mathbb{Z}_q \text{ tel que } A = g^a,\ B = g^b,\ C = g^c,$$

de déterminer si on a la relation $ab \equiv c$ modulo q.

Si on sait résoudre le problème (classique) de Diffie-Hellman dans G alors on sait résoudre le problème décisionnel de Diffie-Hellman. La réciproque n'est pas connue.

Le problème décisionnel de Diffie-Hellman semble difficile dans de nombreux groupes habituellement utilisés en cryptographie. En particulier dans le sous-groupe d'un $(\mathbb{Z}/p\mathbb{Z})^*$ (avec p premier) engendré par un élément d'ordre q, grand premier divisant $p-1$.

1.3. — **Exercice.** Soit p un (grand) premier impair. Le problème de Diffie-Hellman n'est pas (toujours) difficile dans le groupe $(\mathbb{Z}/p\mathbb{Z})^*$.

1.4. — **Théorème.** *Le cryptosystème de El Gamal est sémantiquement sûr sous l'hypothèse que le problème décisionnel de Diffie-Hellman (DDH) est difficile.*

DÉMONSTRATION — Exercice. □

2. Optimal Asymmetric Encryption Padding (OAEP)

Modèle de l'oracle aléatoire

Dans cette section, nous employons les termes de générateur aléatoire et de fonction de hachage dans le contexte du *modèle de l'oracle aléatoire* dont voici l'idée.

Une *fonction de hachage* est une fonction de $\{0,1\}^\ell$ dans $\{0,1\}^k$ avec $k \leqslant \ell$. On parle de *fonction de hachage parfaite* lorsque la structure de cette fonction est inexploitable. Plus précisement, on suppose qu'à chaque instant, la connaissance des empreintes de messages déjà obtenues ne donne aucune information sur les empreintes de messages pour lequel la fonction n'a pas encore été évaluée.

Un *générateur* est une fonction de $\{0,1\}^k$ dans $\{0,1\}^\ell$ avec $k \leqslant \ell$. On parle de *générateur aléatoire* lorsque la structure de cette fonction est inexploitable, avec une signification analogue à celle ci-dessus.

OAEP

Soit un cryptosystème à clé publique pour lequel les espaces $\mathcal{P}$ des messages clairs et $\mathcal{C}$ des messages chiffrés sont des sous-ensembles de $\{0,1\}^n$ (dans la pratique, c'est RSA avec un module de 2^n bits). On note e_K la fonction de chiffrement associée à une clé K. Nous supposerons que cette fonction de chiffement est sûre au sens classique, c'est-à-dire qu'il est en pratique impossible de retrouver la totalité du clair à partir du chiffré (sans connaître la clé de déchiffrement). Soient g un générateur (aléatoire) $\{0,1\}^k \to \{0,1\}^\ell$ et h une fonction de hachage (parfaite) $\{0,1\}^\ell \to \{0,1\}^k$, avec $k+\ell = n$. La construction OAEP consiste à chiffrer les messages $m \in \{0,1\}^\ell$ de la façon suivante.

$$c = \mathcal{E}_K(m) = e_k(m \oplus g(r) \,\|\, r \oplus h(m \oplus g(r))) \qquad \text{où } r \in \{0,1\}^k \text{ est choisi au hasard.}$$

2.1. — Théorème. *Si la fonction de chiffrement e_K est sûre alors le cryptosystème obtenu est sémantiquement sûr dans le modèle de l'oracle aléatoire.*

JUSTIFICATION INTUITIVE — Supposons que c soit le chiffré de l'un de deux messages m_1 ou m_2, c'est-à-dire que

$$c = e_K(m_1 \oplus g(r_1) \,\|\, r_1 \oplus h(m_1 \oplus g(r_1))) \qquad \text{ou} \qquad c = e_K(m_2 \oplus g(r_2) \,\|\, r_2 \oplus h(m_2 \oplus g(r_2)))$$

et qu'un attaquant ait ces trois valeurs en sa possession. Il s'agit de justifier le fait qu'il ne peut pas trancher la réponse à la question "Est-ce que c est le chiffré de m_1 ?". Si la fonction de chiffrement est sûre (au sens faible classique), il ne peut pas obtenir la valeur complète de l'argument de e_K. S'il n'obtient pas la valeur complète de la première partie $m_i \oplus g(r_i)$, il n'aura aucune information sur son haché et donc sur r_i, ainsi que sur $g(r_i)$. Il ne pourra donc pas trancher entre m_1 et m_2. S'il obtient la valeur complète de la première partie, il n'aura pas la valeur complète de la seconde $r_i \oplus h(m_i \oplus g(r_i))$. Mais puisqu'il peut calculer la valeur de h, son manque d'information se reporte sur r_i, ce qui lui interdit d'avoir la moindre idée sur la valeur de $g(r_i)$ et donc de trancher. □

3. Le status du problème de Diffie-Hellman

Différents travaux ont été menés pour cerner la difficulté du problème de Diffie-Hellman, et en particulier pour le relier au problème du logarithme discret. En voici un aperçu.

Bits de poids fort de Diffie-Hellman

On s'intéresse à un sous-groupe G de $(\mathbb{Z}/p\mathbb{Z})^*$, de générateur g. Ici, comme souvent dans ce contexte d'arithmétique modulo p, la notion de *k bits de poids fort* d'un entier x signifie l'entier v tel que

$$v\frac{p}{2^k} \leqslant x \bmod p < (v+1)\frac{p}{2^k}.$$

Autrement dit, ce sont les k bits suivant la virgule dans le développement binaire du rapport x/p.

Le résultat suivant [24] montre en quelque sorte que la difficulté du problème de Diffie-Hellman est concentrée dans les "bits de poids fort".

3.1. — Théorème. Soit $n = \lceil \ln p \rceil$ et $k = \lceil \sqrt{n} \rceil + \lceil \ln n \rceil$. Un oracle donnant les k "bits de poids fort" de g^{xy} étant donnés g^x et g^y permet de résoudre Diffie-Hellman dans G.

PRINCIPE DE LA DÉMONSTRATION — L'oracle permet de trouver un entier t tel que

$$\frac{t}{2^k} \leqslant \frac{g^{xy} \bmod p}{p} < \frac{t+1}{2^k}.$$

En prenant u entier dans l'intervalle $[pt/2^k, p(t+1)/2^k[$, on a une approximation de $g^{xy} \bmod p$:

$$|u - g^{xy} \bmod p| \leqslant \frac{p}{2^k}.$$

Soit $C = g^{ab}$ à calculer, connaissant g^a et g^b. Quitte à remplacer b par $b+s$ pour un s convenable, on peut supposer que $h = g^b$ engendre G. Soit $d = 2\lceil\sqrt{n}\rceil$. Pour r_1, r_2, …, r_d aléatoires dans $[0, |G|-1]$, on interroge l'oracle avec les valeurs g^{a+r_i} et g^b. On obtient des approximations des $v_i = CR_i \bmod p$, avec $R_i = h^{r_i}$:

$$|u_i - v_i| \leqslant \frac{p}{2^k} \qquad \text{pour } 1 \leqslant i \leqslant d.$$

Considérons le réseau L de matrice génératrice

$$\begin{pmatrix} p & 0 & \cdots & 0 & 0 \\ 0 & p & \cdots & 0 & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & \cdots & p & 0 \\ R_1 & R_2 & \cdots & R_d & 1/p \end{pmatrix}.$$

On a $V = (v_1, \ldots, v_d, C/p) \in L$ et $U = (u_1, \ldots, u_d, 0)$ proche de V pour la norme euclidienne. Précisément, $||U - V|| = \sqrt{d+1} \cdot p/2^k$. L'algorithme de Babaï [13] permet de trouver V à partir de U et donc de découvrir C. □

Un oracle Diffie-Hellman pour calculer des logarithmes discrets

Le résultat suivant est dû à den Boer, et il est rappelé dans [80].

3.2. — Théorème. *Soit G un groupe d'ordre q premier et de générateur g. Si $q-1 = \prod_{i=1}^{r} q_i^{e_i}$ est friable (i.e. les facteurs q_i sont petits), alors dans G relativement au générateur g, le problème de Diffie-Hellman est équivalent au problème du logarithme discret.*

Principe de la démonstration — Supposons que l'on ait un oracle $\mathcal{O}$ pour Diffie-Hellman, c'est-à-dire un algorithme tel que $\mathcal{O}(g^u, g^v) = g^{uv}$ pour $u, v \in \mathbb{Z}$, et montrons que l'on peut calculer le logarithme $a \in [0, q-1]$ de $A = g^a \in G$. On a $a = 0$ si et seulement si A est l'élément neutre de G. On peut donc supposer que $a \neq 0$.

Puisqu'on connait la factorisation de $q-1$, on peut trouver facilement un générateur h de $(\mathbb{Z}/q\mathbb{Z})^*$. Soit α le logarithme de $a \equiv h^\alpha$ modulo q en base h. Pour chaque i tel que $1 \leqslant i \leqslant r$, on a

$$a^{\frac{q-1}{q_i}} \equiv h^{\frac{q-1}{q_i}\alpha} \equiv h^{\frac{q-1}{q_i}(\alpha \bmod q_i)} \pmod{q},$$

et donc

$$g^{a^{(q-1)/q_i}} = g^{h^{(\alpha \bmod q_i)(q-1)/q_i}}. \tag{1}$$

L'oracle permet, connaissant les exponentielles de deux puissances de a, d'additionner leurs exposants :

$$\mathcal{O}(g^{a^x}, g^{a^y}) = g^{a^{x+y}} \qquad \text{pour } x, y \in \mathbb{Z}.$$

Il permet donc de calculer l'exponentielle (de base g) de n'importe quelle puissance de a si on l'utilise judicieusement (par exemple, en l'intégrant dans l'un des algorithmes d'exponentiation donnés au chapitre 8). En particulier, on peut calculer l'exponentielle de $a^{(q-1)/q_i}$ de cette façon. On peut aussi calculer les exponentielles des $h^{t(q-1)/q_i}$ pour $0 \leqslant t < q_i$ (de façon classique, l'oracle n'est ici pas nécessaire). On peut alors utiliser (1) pour déterminer $\alpha \bmod q_i$.

En procédant comme dans Pohlig-Hellman (décrit dans le chapitre sur le logarithme discret), on peut aussi déterminer α modulo $q_i^{e_i}$ connaissant α modulo $q_i^{e_i-1}$. Pour cela, poser $a' = ah^{-\alpha \bmod q_i^{e_i-1}} \bmod q$ et $\alpha' = \alpha - (\alpha \bmod q_i^{e_i-1})$. On a alors $a' \equiv h^{\alpha'}$ modulo q et

$$a'^{(q-1)/q_i^{e_i}} \equiv h^{\alpha'(q-1)/q_i^{e_i}} \equiv \left(h^{\frac{q-1}{q_i}}\right)^{\left(\frac{\alpha'}{q_i^{e_i-1}} \bmod q_i\right)} \pmod{q}.$$

En utilisant l'oracle, on peut calculer l'exponentielle de $a'^{(q-1)/q_i^{e_i}}$. En comparant avec les exponentielles des $h^{t(q-1)/q_i}$ (pour $0 \leqslant t < q_i$) déjà calculées, on peut alors déterminer $\alpha'/q_i^{e_i-1}$ modulo q_i, et donc α modulo $q_i^{e_i}$.

Enfin, le théorème chinois permet de déterminer α modulo q et donc a. □

Le coût est, paraît-il, $O(\ln q)$ appels de l'oracle, et $O(\ln^2 qB/\ln B)$ opérations (B étant un majorant des q_i), ce qui ne me paraît pas clair. Mais je conviens d'une complexité en $O(B\ln^2 q)$, appels de l'oracle et opérations de groupe confondues.

Il semble qu'on peut généraliser à un groupe G cyclique d'ordre q non-premier de factorisation connue lorsque $\varphi(q)$ est friable, en utilisant Pohlig-Hellman. De plus, si q admet des facteurs carrés p^2 alors $p \mid \varphi(q)$ et, $\varphi(q)$ étant friable, p est petit.

Equivalence avec le logarithme discret

Le résultat suivant [80] établit plus ou moins l'équivalence entre le problème de Diffie-Hellman et celui du logarithme discret. Il reprend le principe du résultat précédent en remplaçant le groupe $(\mathbb{Z}/q\mathbb{Z})^*$ par celui d'une courbe elliptique. Il n'y a plus qu'à espérer que pour tout groupe d'ordre q premier, on peut trouver une courbe elliptique sur $\mathbb{F}_q$ dont le groupe est cyclique et d'ordre friable, ce qui semble assez raisonnable.

3.3. — Théorème. *Soit G un groupe d'ordre q premier et de générateur g. Supposons qu'on connaisse une courbe elliptique E **cyclique** sur $\mathbb{F}_q$ d'ordre $m = \prod_{i=1}^r q_i^{e_i}$ friable. Alors dans G relativement au générateur g, le problème de Diffie-Hellman est équivalent au problème du logarithme discret.*

PRINCIPE DE LA DÉMONSTRATION — Comme précédemment, on suppose l'existence d'un oracle pour le problème de Diffie-Hellman dans G. Cela permet de faire de "l'arithmétique exponentielle" : l'addition des exposants g^{u+v} se fait par une multiplication $g^u \cdot g^v$ dans G et leur multiplication g^{uv} s'obtient en utilisant l'oracle.

Soit $y^2 = x^3 + \lambda x + \mu$ l'équation de E. Pour $A \in G$ donné, on souhaite calculer $a \in [0, q-1]$ tel que $A = g^a$. On peut là-aussi supposer que $a \neq 0$. De plus, quitte à multiplier A par une puissance de g convenable, on peut supposer que $c = a^3 + \lambda a + \mu$ est un carré modulo q, ce que l'on peut vérifier en calculant $g^{c^{(q-1)/2}}$ à partir de g^a à l'aide de l'oracle et en vérifiant que cela est égal à g. Dans ce cas, on peut ensuite utiliser un algorithme classique pour déterminer, toujours à l'aide de l'oracle, une racine carrée b de c modulo q, ou plus exactement calculer g^b à partir de g^c. Le point $P = (a, b)$ est alors sur E.

Pour $T = (x, y) \in E$, je noterai g^T le couple (g^x, g^y). Soit H un générateur de E (puisque m est friable, un tel générateur est facile à expliciter) et α le logarithme de P, c'est-à-dire $P = \alpha H$. Pour chaque i tel que $1 \leqslant i \leqslant r$ on a

$$\frac{m}{q_i}P = \alpha\frac{m}{q_i}H = (\alpha \bmod q_i)\frac{m}{q_i}H,$$

donc

$$g^{(m/q_i)P} = g^{(\alpha \bmod q_i)(m/q_i)H}.$$

Pour déterminer α modulo q_i, il suffit donc de calculer les couples $g^{t(m/q_i)H}$ pour $0 \leqslant t < q_i$, et $g^{(m/q_i)P}$ puis comparer. Calculer les $g^{t(m/q_i)H}$ se fait de façon classique, en utilisant les formules d'addition sur la courbe. Pour obtenir $g^{(m/q_i)P}$ (rappelons que P n'est pas connu, mais que g^P l'est), il suffit de savoir calculer $g^{P_1+P_2}$ à partir de g^{P_1} et g^{P_2} pour $P_1, P_2 \in E$. Les coordonnées homogènes de $P_1 + P_2$ s'expriment polynomialement en fonction de celles de P_1 et P_2. Le calcul de $g^{P_1+P_2}$ est donc encore une application de "l'arithmétique exponentielle" permise par l'oracle.

Il ne reste plus qu'à faire comme dans le résultat précédent pour obtenir la valeur de α modulo les $q_i^{e_i}$, puis modulo m, et calculer P pour en extraire a. □

Couplages et Diffie-Hellman

Soit $P \in E[\ell]$ un point d'ordre ℓ tel que $\langle P, P\rangle \neq 1$ (il ne faudra donc pas choisir le couplage de Weil !). Le couplage bilinéaire peut servir à résoudre une instance du problème Décisionnel de Diffie-Hellman elliptique

de base P. Soit en effet (P, aP, bP, cP) une telle instance (le problème est alors de déterminer si $c \equiv ab$ modulo l). On a alors

$$\langle aP, bP\rangle = \langle P, P\rangle^{ab} \quad \text{et} \quad \langle P, cP\rangle = \langle P, P\rangle^{c}.$$

Si l est premier, il suffit donc de comparer les deux membres de droite pour décider du statut de l'instance.

3.4. — Exercice. Montrer qu'un couplage bilinéaire peut fournir un protocole d'échange de clés entre trois parties. Indication : utiliser l'identité

$$\langle aP, bP\rangle^{c} = \langle cP, aP\rangle^{b} = \langle bP, cP\rangle^{a}.$$

Sur quel problème sa sécurité repose-t-elle ?

Chapitre 12

Signature, authentification

La cryptographie ne se limite plus à l'art de chiffrer. On va considérer dans ce chapitre et le suivant de nouvelles tâches qu'il est possible de réaliser sous forme numérique.

• L'*identification*. Elle permet à quelqu'un de prouver qu'il est celui qu'il prétend être, ou qu'il a le droit d'accéder à un service. C'est la fonction réalisée (primitivement) lors d'une connexion sur un ordinateur, à l'aide d'un mot de passe. C'est aussi celle réalisée lorsque l'on présente une carte bancaire, et que l'on tape son code secret.

• L'*authentification*. Destinée à garantir l'authenticité d'un document lors d'une transmission. Le destinataire pourra vérifier que le document qu'il reçoit est absolument identique au document original, qu'il n'aura pas été altéré (intentionnellement ou non) pendant la transmission (les techniques de codage ne permettent de se garantir que contre les altérations non intentionnelles). Contrairement à une signature, il se peut que seul le destinataire soit en mesure de vérifier l'authenticité.

• La *signature* proprement dite. C'est l'analogue de la signature manuscrite, que l'on appose par exemple au bas d'un contrat, ou sur un chèque. Elle engage la responsabilité du signataire, qui ne pourra pas la renier plus tard. La signature ne pourra pas être imitée (sauf avec une probabilité infime) et pourra être vérifiée par n'importe qui.

0.1. — Définition. *Un procédé de signature est la donnée de*

• *Un ensemble fini $\mathcal{P}$, appelé l'espace des messages*

• *Un ensemble fini $\mathcal{S}$, appelé l'espace des signatures*

• *Un ensemble fini $\mathcal{K}$, appelé l'espace des clés*

• *Pour chaque clé $k \in \mathcal{K}$, une fonction de signature $s_k : \mathcal{P} \to \mathcal{S}$ et une fonction de vérification $v_k : (\mathcal{P}, \mathcal{S}) \to \{\texttt{oui}, \texttt{non}\}$ telles que $v_k(x, s_k(x)) = \texttt{oui}$ pour tout $x \in \mathcal{P}$.*

Dans la pratique, il y a deux types de schémas de signature.

• Les schémas de signature *avec restauration du message*. Dans ce cas, la signature est une donnée dont le traitement permet de retrouver le message et de garantir son authenticité. Le message clair n'a donc pas à être transmit en plus de la signature. Par contre, la signature est nécessairement au moins aussi longue que le message.

• Les schémas de signature *en appendice*. La signature est alors en général une valeur beaucoup plus courte que la donnée qu'elle authentifie. Pour la vérifier, il faut disposer de la signature et du message signé (que l'on a pu obtenir ensemble ou séparément).

1. Signature RSA

Le choix des paramètres p, q, N, E, d se fait de la même façon que pour chiffrer mais il est fait par l'expéditeur Alice, qui diffuse sa clé publique (N, E). Il peut d'ailleurs très bien se servir de la même clé pour recevoir des messages chiffrés que pour envoyer des messages signés.

Pour signer un message $m \in \mathbb{Z}_N$ qu'elle veut envoyer à Bob, Alice calcule $s = m^d \bmod N$. Dans son envoi à Bob, elle fait accompagner m par sa signature s. A la réception, Bob peut contrôler la signature d'Alice en vérifiant que $s^E \bmod N = m$. S'il y a bien égalité, Bob est assuré que le message m provient bien d'Alice et qu'il n'a pas été altéré puisqu'elle est la seule à posséder la clé secrète d nécessaire pour calculer s.

Il est clair que n'importe qui peut calculer des couples (m, s) de messages apparemment signés par Alice en choisissant d'abord s et en calculant $m = s^E \bmod N$. Mais le message produit m est imprévisible et n'a pratiquement aucune chance de représenter une message intelligible. Toutefois, la possibilité ce genre d'attaque (appelée contrefaçon existentielle) peut être dans de nombreux contextes un problème grave. Par exemple, la gestion à grande échelle de clés publiques se fait à l'aide de certificats, c'est-à-dire de signatures de

clés publiques (par une autre clé dont on est sûr de l'authenticité). Puisque presque n'importe quelle valeur numérique peut représenter une clé publique il est indispensable dans ce contexte que toute contrefaçon existentielle soit pratiquement impossible.

Schémas de redondance

Afin de se protéger contre les attaques existentielles et aussi de certains inconvénients dus à la propriété multiplicative de RSA, on utilise des schémas de redondance qui précisent une façon dont doivent être formatée les messages à signer avant de leur appliquer l'exponentiation RSA. Le plus connu de ces schémas de redondance est la norme ISO-9796 dont l'élaboration à été perturbée par l'apparition de nombreuses attaques sur les versions successives de cette norme. A l'heure actuelle, il est encore difficile de d'affirmer que la dernière version de cette norme ne subira plus d'attaque majeure.

2. Signatures El Gamal, DSS et EC-DSA

Ces trois procédés de signature sont semblables et leur sécurité est apparentée au problème du logarithme discret. Ils diffèrent essentiellement par le groupe utilisé.

Signature El Gamal

L'expéditeur Alice choisit un (grand) nombre premier p ainsi qu'une racine primitive g modulo p. Il choisit aussi un entier a dans $[1, p-2]$ et calcule $A = g^a \bmod p$. Il publie les entiers p, g et A mais garde a secret.

Pour signer chaque message $m \in \mathbb{Z}_p$, Alice choisit un entier k dans $[1, p-2]$ tel que $\operatorname{pgcd}(k, p-1) = 1$ et calcule $K = g^k \bmod p$. Elle calcule aussi $s = (m - aK)k^{-1} \bmod (p-1)$ et, dans son envoi à Bob, elle fait accompagner le message m par sa signature (K, s).

A la réception, Bob s'assure que la signature est bien celle d'Alice en vérifiant que $1 \leqslant K \leqslant p-1$ et $A^K K^s \equiv g^m$ modulo p. En effet

$$A^K K^s \equiv (g^a)^K (g^k)^{(m-aK)k^{-1}} \equiv g^{aK} g^{m-aK} \equiv g^m \pmod{p}.$$

Si Bob néglige de vérifier la condition $1 \leqslant K \leqslant p-1$, il peut être dupé par Oscar si celui-ci est en possession d'un message m et de la signature (K, s) d'Alice correspondante. En effet, pour le message m' de son choix, Oscar peut calculer (à condition que $\operatorname{pgcd}(m, p-1) = 1$)

$$u = m'm^{-1} \bmod (p-1), \qquad K' \text{ tel que } K' \equiv \begin{cases} Ku \pmod{p-1} \\ K \pmod{p} \end{cases}, \qquad s' = su \bmod (p-1).$$

L'égalité $A^{K'} K'^{s'} \equiv g^{m'}$ modulo p est alors vérifiée.

Pour que le procédé de signature soit sûr, il est nécessaire de prendre les mêmes précautions que pour le chiffrement El Gamal, par exemple dans le choix du nombre premier p et dans la non-réutilisation du nombre k.

Sécurité de la signature El Gamal

Un procédé de signature est sûr s'il est difficile de contrefaire une signature. Comment Oscar peut-il s'y prendre, s'il veut contrefaire la signature d'Alice sur un message m, sans connaître a ? S'il commence par choisir K, il doit ensuite trouver s vérifiant $K^s \equiv A^{-K} g^m$ modulo p c'est-à-dire résoudre une instance du logarithme discret. Toutefois, il peut envisager de commencer par choisir s et ensuite chercher K vérifiant $A^K K^s \equiv g^m$ modulo p. Ce problème n'est plus un logarithme discret et est moins bien connu. Oscar peut aussi envisager une attaque plus générale qui ne commencerait pas par un choix de K ou de s. Bref, la sécurité du procédé de signature El Gamal est mal connue. Toutefois, on pense généralement (à tort ou à raison) qu'elle est équivalente à celle du logarithme discret.

Par un procédé moins trivial que pour RSA, Oscar peut fabriquer des messages imprévisibles (c'est-à-dire qu'il ne peut pas choisir ces messages) portant la signature d'Alice. Pour cela, il lui suffit de choisir deux entiers i, j dans $[0, p-2]$ tels que $\operatorname{pgcd}(j, p-1) = 1$ et de calculer

$$K = g^i A^j \pmod{p}, \qquad s = -Kj^{-1} \pmod{p-1}, \qquad m = si \pmod{p-1}.$$

Le couple (K, s) est alors une signature valide du message m.

D.S.S.

C'est une amélioration du schéma de El Gamal. A sa création, en 1991, il s'est tout d'abord appelé D.S.A. (Digital Signature Algorithm). Son nom est devenu D.S.S. (Digital Signature Standard) lorsqu'il est devenu une norme fédérale en 1994.

Pour une sécurité identique, une signature D.S.S. est plus courte qu'une signature El Gamal. L'idée directrice est la suivante. On connaît désormais des algorithmes pour résoudre le logarithme discret dans $(\mathbb{Z}/p\mathbb{Z})^*$ (algorithmes sous-exponentiels) qui, bien que n'étant pas polynomiaux, ont repoussé les limites des p accessibles au logarithme discret. Toutefois, ces algorithmes ne s'appliquent pas à des groupes quelconques, pour lesquels on ne dispose encore que d'algorithmes exponentiels. Dans le schéma D.S.S., on travaille dans un sous-groupe d'ordre q de $(\mathbb{Z}/p\mathbb{Z})^*$ où p est un nombre premier de 512 bits et q un de 160 bits. Un attaquant éventuel doit, s'il veut utiliser une méthode sous-exponentielle, travailler sur 512 bits. Sinon, il peut travailler sur 160 bits mais dispose dans ce cas seulement des méthodes exponentielles.

Puisque la sécurité assurée par un nombre p de 512 bits est devenue incertaine, le standard initial a été modifié pour permettre jusqu'à 1024 bits, par incréments de 64 bits.

Les messages que peut signer D.S.S. ont pour longueur 160 bits. Nous verrons plus loin comment obtenir de tels messages de longueur fixe à partir de messages réels, en utilisant une fonction de hachage. Le standard précise que la fonction de hachage utilisée est SHA-1.

L'expéditeur Alice choisit un nombre premier q de 160 bits, puis un nombre premier p de $512 + 64t$ bits $(0 \leqslant t \leqslant 8)$ tel que $q \mid p-1$. Alice choisit aussi un entier g dans $[1, p-1]$ d'ordre q modulo p. Enfin, Alice choisit un entier a dans $[1, q-1]$ et calcule $A = g^a \bmod p$. Elle publie les entiers p, q, g et A mais garde a secret.

Pour signer chaque message $m \in \mathbb{Z}_q$, Alice choisit un autre entier k dans $[1, q-1]$ et calcule la valeur $K = (g^k \bmod p) \bmod q$. Elle calcule aussi $s = (m + aK)k^{-1} \bmod q$ et, dans son envoi à Bob, elle fait accompagner le message m par sa signature (K, s).

A la réception, Bob s'assure que la signature est bien celle d'Alice en vérifiant que $1 \leqslant K, s \leqslant q$ et que $(A^{Ks^{-1}} g^{ms^{-1}} \bmod p) \bmod q = K$, où s^{-1} désigne l'inverse de s modulo q. En effet, de manière analogue au schéma de El Gamal (mais un signe $-$ est devenu un signe $+$ dans D.S.S.) on a

$$A^K g^m \equiv (g^k)^s \pmod{p} \qquad \text{donc} \qquad A^{Ks^{-1}} g^{ms^{-1}} \bmod p = (g^k) \bmod p.$$

En réduisant à nouveau modulo q, on obtient la relation vérifiée par Bob.

Tout cela suppose que K et s sont non nuls, mais il est hautement improbable que l'un des deux le soit. Eventuellement, Alice peut s'assurer qu'ils sont non nuls et changer de k sinon. Le schéma 1 résume le procédé de signature DSS.

D.S.A. sur courbes elliptiques

On peut utiliser le groupe d'une courbe elliptique pour définir un protocole de signature. Le schéma 2 représente le protocole EC-DSA tel qu'il est proposé par le rapport IEEE/P1363. Ici, X_K est l'entier déterminé par la représentation binaire de l'abscisse de K. De même pour X_P. Comme dans DSS, il faut que c et s soient non nuls. Si ce n'est pas le cas, choisir un autre k. Dans la vérification, s^{-1} désigne l'inverse de s modulo r.

3. Fonctions de hachage en signature

Les procédés de signature ne permettent en général de signer que des petits messages, de longueur fixe (typiquement 128 ou 160 bits). Dans la pratique, on souhaite pouvoir signer des messages beaucoup plus longs. Une méthode naïve consiste à découper le message long en blocs, et à signer chacun des blocs. Cette approche a toutefois plusieurs inconvénients : la signature complète est très grande, son élaboration est très lente et elle ne garantit pas contre un ré-arrangement des blocs qui peut changer la signification du message.

Une meilleure solution consiste à utiliser une fonction de hachage pour calculer, à partir d'un message arbitrairement long, une empreinte de longueur fixée (penser aux empreintes digitales qui, bien que ne

Paramètres :	q (premier de 160 bits)		public
	$p \equiv 1 \bmod q$ (premier de $512 + 64t$ bits)		public
	g entier d'ordre q modulo p		public
	$a \in [1, q-1]$		secret
	$A = g^a \bmod p$		public
	Alice		**Bob**
Signature :	$k \in [1, q-1]$		
	$K = (g^k \bmod p) \bmod q$		
	$s = (m + aK)k^{-1} \bmod q$	$\overset{m,K,s}{\longrightarrow}$	
Vérification :			$1 \leqslant K, s \leqslant q$?
			$(A^{Ks^{-1}} g^{ms^{-1}} \bmod p) \bmod q \overset{?}{=} K$

Schéma 1. Signature DSS

Paramètres :	Corps fini $\mathbb{F}_q$		public
	Courbe elliptique E sur $\mathbb{F}_q$		publique
	G point de E d'ordre r, grand premier		publics
	$a \in [1, r-1]$		secret
	$A = aG$		public
	Alice		**Bob**
Signature :	$k \in [1, r-1]$		
	$K = kG$		
	$t = X_K \bmod r$		
	$s = (m + at)k^{-1} \bmod r$	$\overset{m,t,s}{\longrightarrow}$	
Vérification :			$1 \leqslant t, s \leqslant r-1$?
			$P = ms^{-1}G + ts^{-1}A$
			$X_p \bmod r \overset{?}{=} t$

Schéma 2. Signature ECDSA

représentant qu'une information très partielle sur son propriétaire, permettent de l'indentifier avec quasi-certitude). Ensuite, on signe l'empreinte pour authentifier le message.

Un attaquant éventuel en possession d'un message x, de son empreinte et de sa signature s peut, s'il est capable de calculer un autre message x' de même empreinte, faire croire que le message x' est authentique en l'accompagnant de la signature s. Pour contrer cela, il faut que la fonction de hachage soit (au moins) faiblement résistante aux collisions. De plus, un attaquant cherchant à obtenir d'Alice la signature d'un

message qu'elle ne signerait pas de son plein gré peut envisager la stratégie suivante. Il génère un grand nombre de variantes du message qui l'intéresse (il peut varier le nombre d'espaces, les coupures des lignes, l'ordre de certains mots ou certaines phrases et changer de formulation) et calcule les empreintes de chacune d'elles. Il génère aussi un grand nombre de messages qu'Alice est susceptible d'accepter de signer, et il calcule leurs empreintes. S'il obtient un message x_2 de la deuxième catégorie ayant la même empreinte qu'un x_1 de la première catégorie, il peut faire signer x_2 par Alice puis utiliser la signature obtenue pour accompagner x_1. Pour contrer cette attaque, la fonction de hachage doit être (fortement) résistante aux collisions.

Chapitre 13

Identification

Ce chapitre s'intéresse à une application cryptographique qui a pris une ampleur majeure avec l'essor de la cryptographie moderne. L'identification est un procédé par lequel une entité (personne ou machine) prouve qui elle est afin d'accéder à un service auquel elle à droit.

Tous les procédés d'identification font intervenir une donnée secrète, censée être connue seulement de la personne autorisée. Pour s'identifier, celle-ci doit prouver qu'elle connaît cette donnée secrète.

1. Protocoles à une passe

Mots de passe

Dans les procédés d'identification à mot de passe, la personne autorisée prouve qu'elle connaît le secret en transmettant celui-ci (ou une valeur calculée directement à partir de celui-ci). C'est la méthode employée dans sa version la plus simple (et la moins sûre) dans le système `Unix` ou lors d'une transaction utilisant une carte bancaire à puce. Sa principale faiblesse est l'attaque par répétition (*replay*) : il suffit à un attaquant de capter le mot de passe au moment où il est utilisé pour pouvoir ensuite se faire passer pour la personne autorisée (c'est particulièrement facile si l'attaquant est le vérificateur lui-même).

Schéma de Lamport

Le *schéma de Lamport* permet de contrer cette attaque, mais ne permet qu'un nombre fini t fixé à l'avance de connexions successives. Il utilise une fonction h à sens unique (une bijection de l'espace des mots de passe dans lui-même, telle que $h(x)$ soit facile à calculer connaissant x mais telle qu'on ne sache pas calculer x à partir de $h(x)$). Dans ce schéma, si Alice doit s'identifier auprès de Bob, un mot w de l'espace des mots de passe est auparavant choisi par Alice et elle transmet $w_0 = h^t(w)$ à Bob. Lorsque elle souhaite s'identifier pour la i-ème fois ($1 \leqslant i \leqslant t$), Alice (ou sa machine) calcule $w_i = h^{t-i}(w)$ et transmet la valeur calculée à Bob. Bob accepte l'identité d'Alice si $h(w_i) = w_{i-1}$. Bob mémorise alors w_i à la place de w_{i-1} en prévision de la prochaine identification.

2. Protocoles à deux passes

Identification à clé publique par challenge

On utilise un cryptosystème asymétrique ou un procédé de signature. Le secret d'Alice est sa clé privée. Elle démontre la connaissance de ce secret en signant ou en déchiffrant une valeur aléatoire proposée par le serveur. On note Enc, Dec, Sign les fonctions de chiffrement, de déchiffrement et de signature.

Alice		**Bob**
	$\xleftarrow{m}$	aléa $m \in \mathcal{P}$
$s = \mathrm{Sign}(m)$	$\xrightarrow{s}$	s signature valide de m ?

Protocole 1. Identification par signature

Alice		**Bob**
	$\xleftarrow{c}$	aléa $m \in \mathcal{P}$, $c = \mathrm{Enc}(m)$
$m' = \mathrm{Dec}(c)$	$\xrightarrow{m'}$	$m' \stackrel{?}{=} m$

Protocole 2. Identification par déchiffrement

3. Protocoles sans divulgation d'information (*Zero-knowledge*)

Le protocole de Fiat-Shamir

Ce protocole utilise un entier $n = pq$ tel ceux utilisés pour RSA fourni par un tiers de confiance. Le secret détenu par Alice est la racine carrée a modulo n d'un entier public $A \in [0, n-1]$ ("public" peut vouloir dire par exemple que A est associé au nom d'Alice dans un annuaire certifié par le tiers de confiance). Alice est identifiée auprès de Bob au moyen des trois échanges suivants.

• Alice choisit un entier aléatoire k (l'*engagement*), dans l'intervalle $[0, n-1]$ et calcule $K = k^2 \bmod n$. Elle transmet K à Bob.

• Bob choisit un booléen r (le *défi*) au hasard (0 ou 1) et le transmet à Alice.

• Alice calcule $y = ka^r \bmod n$ (la *réponse*) et le transmet à Bob. Bob vérifie que $y^2 \equiv KA^r$ modulo n.

Alice		**Bob**
$K = k^2 \bmod n$	$\xrightarrow{K}$	
	$\xleftarrow{r}$	$r \in \{0, 1\}$
$y = ka^r \bmod n$	$\xrightarrow{y}$	$y^2 \stackrel{?}{\equiv} KA^r \pmod n$

Protocole 3. Identification par Fiat-Shamir

Ce protocole est *consistant* (*completness property*), ce qui signifie que la connaissance du secret a permet à Alice de répondre au défi proposé par Bob, quelle que soit sa valeur.

Ce protocole est *significatif* (*soundness*) c'est-à-dire que, pour réussir son identification avec probabilité acceptable (ici supérieure à 1/2), Alice doit connaître le secret a. En effet, Alice ne peut pas prévoir la valeur de r au moment où elle transmet K à Bob. Au risque d'échouer dans son identification, Alice doit être en mesure de fournir à Bob les deux réponses possibles k et ka de y (mais elle n'en fournira qu'une pour ne pas dévoiler son secret) et donc doit connaître a.

Un attaquant éventuel souhaitant se faire passer pour Alice mais ne connaissant pas a peut tenter de choisir k et de transmettre $K = k^2 \bmod n$ à Bob. Dans le cas où $r = 0$, il pourra alors fournir à Bob la valeur de $y = k$, mais sera bloqué si $r = 1$. Autre alternative, l'attaquant peut tenter de choisir k_1 au hasard et de transmettre $K = k_1^2/A \bmod n$ à Bob. Dans le cas où $r = 1$ il pourra alors fournir à Bob la racine carrée k_1 de KA, mais sera bloqué si $r = 0$. Au mieux, l'attaquant dupera Bob au plus une fois sur deux. En acceptant l'identification seulement après plusieurs itérations de ces trois échanges, on peut rendre la probabilité de duper Bob arbitrairement faible.

Le protocole de Schnorr

Ce protocole utilise deux nombres premiers p, q tels que $q \mid p-1$ et un entier g d'ordre q modulo p. Ces données sont fournies par un tiers de confiance et sont choisies de telle sorte que le logarithme discret dans $(\mathbb{Z}/p\mathbb{Z})^*$ soit difficile. Le secret détenu par Alice est un entier $a \in [0, q-1]$ et la donnée $A = g^{-a} \bmod p$ est rendue publique. Alice s'identifie auprès de Bob au moyen des trois échanges suivants.

• Alice choisit un engagement aléatoire k dans l'intervalle $[0, q-1]$ et calcule $K = g^k \bmod p$. Elle transmet K à Bob.

• Bob choisit un défi r au hasard dans l'intervalle $[0, q-1]$ et le transmet à Alice.

• Alice calcule la réponse $y = k + ar \bmod q$ et la transmet à Bob. Bob vérifie que $g^y A^r \equiv K$ modulo p.

Alice		**Bob**
$k \in [0, q-1],\ K = g^k \bmod p$	$\xrightarrow{K}$	
	$\xleftarrow{r}$	$r \in [0, q-1]$
$y = k + ar \bmod q$	$\xrightarrow{y}$	$g^y A^r \stackrel{?}{\equiv} K \pmod p$

Protocole 4. Identification par Schnorr

De la même manière que pour le protocole de Fiat-Shamir, on montre facilement que le protocole de Schnorr est consistant, c'est-à-dire que la connaissance du secret a permet à Alice de fournir à Bob une réponse y valide, quelle que soit la valeur du défi. On montre aussi facilement que le protocole de Schnorr est significatif, c'est-à-dire qu'une personne ne connaissant pas le secret ne peut pas se faire passer pour Alice, sauf avec une probabilité négligeable $(1/q)$.

Pour obtenir un niveau de sécurité satisfaisant avec le protocole de Fiat-Shamir, on est obligé de l'itérer plusieurs fois. Avec l itérations, la probabilité qu'un attaquant réussisse à se faire passer pour Alice est 2^{-l}. Un avantage du protocole de Schnorr sur le protocole de Fiat-Shamir est qu'un niveau de sécurité équivalent est obtenu avec une seule itération. Il suffit pour cela que $q \geqslant 2^l$.

Le protocole de Guillou-Quisquater

Les protocoles de Fiat-Shamir et Schnorr reposent sur les même principes (en particulier les trois phases engagement/défi/réponse). Leur différence essentielle tient à la fonction (homomorphisme de groupes, en fait) à sens unique utilisée (Rabin pour Fiat-Shamir et exponentielle discrète pour Schnorr). Le protocole de Guillou-Quisquater reprend les mêmes principes à l'aide d'une exponentiation RSA. Ce protocole est aujourd'hui largement utilisé dans les cartes à puce.

Une Autorité de Confiance fournit un module RSA n et un exposant de chiffrement E publics. Ces données peuvent être utilisées conjointement par plusieurs couples identifié/identifiant (Alice/Bob). L'Autorité de Confiance garde les facteurs p, q de n et l'exposant de déchiffrement $e = E^{-1} \bmod \varphi(n)$ secrets.

Chaque client Alice possède un secret a et sa contrepartie publique $A = a^{-E} \bmod n$ cette dernière étant certifiée par l'Autorité de Confiance, par un procédé de signature quelconque.

Alice est identifiée auprès de Bob au moyen des trois échanges suivants.

• Alice choisit un engagement $k \in \mathbb{Z}_n^*$ et envoie $K = k^E \bmod n$ à Bob, ainsi que sa clé publique A signée par l'autorité.

• Bob vérifie la signature apposée sur A par l'autorité. Puis il choisit un défi $r \in [0, E-1]$ au hasard et l'envoie à Alice.

• Alice calcule la réponse $y = ka^r \bmod n$ et l'envoie à Bob. Celui-ci vérifie alors que $y^E A^r \equiv K$ modulo n.

Alice		**Bob**
$k \in \mathbb{Z}_n^*,\ K = k^E \bmod n$	$\xrightarrow{K,A}$	
	$\xleftarrow{r}$	$r \in [0, E-1]$
$y = ka^r \bmod n$	$\xrightarrow{y}$	$y^E A^r \stackrel{?}{\equiv} K \pmod n$

Protocole 5. Identification par Guillou-Quisquater

3.1. — Exercice. Montrer que le protocole de Guillou-Quisquater est significatif, sous l'hypothèse que le problème RSA est difficile à résoudre dans le sous-groupe de $(\mathbb{Z}/N\mathbb{Z})^*$ engendré par A.

Non-divulgation d'information

Dans le protocole de Fiat-Shamir, Alice prouve qu'elle connaît un secret a sans le dévoiler. On peut se demander si Bob ou un observateur peut exploiter les données recueillies lors d'identifications successives pour déterminer a. On montre que ce n'est pas le cas, en faisant une *simulation* d'Alice :

Cette simulation est réalisée par une machine hypothétique ne possédant pas le secret a et fonctionnant ainsi :

• Elle tente de deviner (mais elle ne dispose d'aucun moyen pour réussir avec probabilité $> 1/2$) le défi r que va choisir Bob. Elle construit K comme expliqué plus haut de façon à être en mesure de fournir le y correspondant à cette hypothèse sur r. Elle transmet K à Bob.

• Bob choisit le défi r suivant la stratégie habituelle et le transmet à Alice.

• Si le défi r choisi par Bob se révèle être celui prévu par la machine alors celle-ci transmet à Bob la réponse y correspondante. Sinon, la machine ne peut pas continuer et les données fournies lors de cette passe seront écartées car incomplètes.

Supposons que Bob soit en mesure d'exploiter les données accumulées au cours de l identifications réelles d'Alice pour déterminer son secret a. La simulation permet d'obtenir une contradiction. En effet, Bob peut simuler Alice en utilisant la machine décrite ci-dessus. Pour obtenir des données exploitables de l passes, il lui faudra interroger le simulateur environ $2l$ fois. Puisque les données recueillies sont identiques à celles qui auraient pu être recueillies lors d'identifications réelles, Bob pourra aussi les exploiter pour calculer a. En conclusion, Bob pourrait calculer a en utilisant un simulateur ne disposant d'aucune information sur a. C'est la contradiction cherchée.

Si on adapte le même raisonnement au protocole de Schnorr (avec une seule passe mais $q \geqslant 2^l$), on note que Bob doit faire $q \geqslant 2^l$ tentatives avec le simulateur pour obtenir une transaction complète (car la probabilité de deviner r n'est plus que de $1/q$). Le coût de l'utilisation du simulateur est donc exponentiel par rapport au coût d'identifications réelles. Ce raisonnement ne constitue plus, dans ce cas, une preuve satisfaisante de non-divulgation d'information.

3.2. — Exercice. En supposant que E est un nombre premier constant (i.e. indépendant de N), justifier que le protocole de Guillou-Quisquater vérifie la propriété de non-divulgation d'information.

Le protocole d'Okamoto

Le protocole d'Okamoto est un dérivé du protocole de Schnorr, pour lequel on sait prouver qu'il respecte la propriété de non-divulgation d'information. Il utilise aussi deux nombres premiers p et q tels que $q \mid p-1$ et deux entiers α_1, α_2 d'ordre q modulo p (le logarithme de α_1 en base α_2 est inconnu de tous, y compris d'Alice). Le secret d'Alice est la donnée de deux entiers $a_1, a_2 \in [0, q-1]$ et la donnée $A = \alpha_1^{-a_1}\alpha_2^{-a_2} \bmod p$ est rendue publique. Alice s'identifie auprès de Bob au moyen des trois échanges suivants.

• Alice choisit deux entiers aléatoires $k_1, k_2 \in [0, q-1]$ et calcule $K = \alpha_1^{k_1}\alpha_2^{k_2} \bmod p$. Elle transmet K à Bob.

• Bob choisit un entier e au hasard dans l'intervalle $[0, q-1]$ et le transmet à Alice.

• Alice calcule $y_1 = k_1 + a_1 e \bmod q$ et $y_2 = k_2 + a_2 e \bmod q$ et transmet le couple (y_1, y_2) à Bob. Bob vérifie que $K \equiv \alpha_1^{y_1}\alpha_2^{y_2}A^e$ modulo p.

Alice		**Bob**
$k_1, k_2 \in [0, q-1],\ K = \alpha_1^{k_1}\alpha_2^{k_2} \bmod p$	$\xrightarrow{K}$	
	$\xleftarrow{e}$	$e \in [0, q-1]$
$y_1 = k_1 + a_1 e \bmod q, y_2 = k_2 + a_2 e \bmod q$	$\xrightarrow{y_1, y_2}$	$\alpha_1^{y_1}\alpha_2^{y_2}A^e \stackrel{?}{\equiv} K \pmod p$

Protocole 6. Identification par Okamoto

4. Engagement

L'engagement (*commitment*) est une primitive cryptographique qui intervient dans de nombreux protocoles élaborés. Une façon de présenter cette primitive est la suivante. La partie qui s'engage écrit une information (par exemple un bit au hasard) sur un bulletin et place le bulletin dans une boîte opaque. Après que le protocole ait fait intervenir une autre partie, la boîte est ouverte et son contenu dévoilé. Il y a deux aspect qui caractérisent cette primitive.

1) L'information est dissimulée (*concealing property*) dans un premier temps, jusqu'au moment ou elle est révélée.

2) L'information ne peut pas être modifiée (*binding property*) entre le début de l'engagement (le moment où on ferme la boîte) et sa révélation (le moment où on ouvre la boîte).

Deux protocoles similaires

Voici deux protocoles d'engagement d'un bit utilisant les résidus quadratiques. Soit $N = pq$ en entier RSA (le produit de deux grands nombres premiers distincts). On note

$$\mathbb{Z}_N^+ = \{x \in \mathbb{Z}_N \mid (x/N) = 1\}$$
$$Q = \{x^2 \bmod N \mid x \in \mathbb{Z}_N,\ \mathrm{pgcd}(x, N) = 1\} \subseteq \mathbb{Z}_N^+.$$

Dans le premier protocole, Alice est la seule à connaître la factorisation de N. Elle génère un élément m de $\mathbb{Z}_N^+ \setminus Q$, qu'elle rend public. Pour engager un bit $b \in \{0,1\}$, Alice transmet $e = m^b r^2 \bmod N$ où r est un élément aléatoire de $\mathbb{Z}_N^*$. Pour dévoiler b à Bob, elle lui indiquera les valeurs de b et de r et celui-ci pourra alors vérifier que $e \equiv m^b r^2$ modulo N.

	Alice		**Bob**
Paramètres :	$N = pq$, $m \in \mathbb{Z}_N^+ \setminus Q$	$\xrightarrow{N,m}$	
Engagement :	$b \in \{0,1\}$, $r \in \mathbb{Z}_N^*$, $e = m^b r^2 \bmod N$	$\xrightarrow{e}$	
Révélation :		$\xrightarrow{b,r}$	$e \stackrel{?}{\equiv} m^b r^2 \pmod N$

Protocole 7. Engagement utilisant les résidus quadratiques

Dans le deuxième protocole, Bob est le seul à connaître la factorisation de N. Alice génère un élément m de Q, qu'elle rend public. Le protocole est autrement identique au précédent. Pour engager un bit $b \in \{0,1\}$, Alice transmet $e = m^b r^2 \bmod N$ où r est un élément aléatoire de $\mathbb{Z}_N^*$. Pour dévoiler b à Bob, elle lui indiquera les valeurs de b et de r et celui-ci pourra alors vérifier que $e \equiv m^b r^2$ modulo N.

	Alice		**Bob**
Paramètres :		$\xleftarrow{N,m}$	$N = pq$, $m \in Q$
Engagement :	$b \in \{0,1\}$, $r \in \mathbb{Z}_N^*$, $e = m^b r^2 \bmod N$	$\xrightarrow{e}$	
Révélation :		$\xrightarrow{b,r}$	$e \stackrel{?}{\equiv} m^b r^2 \pmod N$

Protocole 8. Engagement utilisant les résidus quadratiques

On notera que dans le premier protocole, la propriété de dissimulation repose sur des arguments de complexité (le bit engagé n'est dissimulé à Bob que s'il dispose d'une puissance de calcul finie) et la valeur du bit est inconditionnellement non-modifiable (la valeur est est non-modifiable par Alice seulement si celle-ci ne dispose que d'une puissance de calcul finie).

Au contraire, dans le deuxième protocole, la propriété de dissimulation est inconditionnelle mais le fait que la valeur du bit soit non-modifiable repose sur des arguments de complexité.

5. Kerberos

Kerberos est une méthode de distribution de clés de session, utilisant les outils de la cryptographie classique. Elle permet par exemple à une entité A (le client Alice) désirant communiquer avec B (le serveur Bob) d'obtenir une clé symétrique qu'elle partagera avec B pour communiquer avec lui de manière sûre. En outre et surtout, Kerberos permet à A de s'identifier auprès de B, c'est-à-dire que B sera sûr que c'est avec A qu'il communiquera avec la clé de session générée. En option, A peut aussi identifier B. Kerberos utilise une autorité de confiance C avec laquelle chaque utilisateur U partage à priori une clé symétrique K_U.

Le développement de Kerberos [97] à commencé en 1978, à l'initiative de DEC et IBM. Il a progressivement évolué et la version actuelle (version 5) date de 1994.

Protocole

Le protocole (simplifié) se déroule de la façon suivante.

• (Demande de ticket.) Le client A s'adresse à C dans le but d'obtenir une clé de session en précisant sa propre identité et celle du serveur B avec lequel il veut communiquer. Il accompagne ces indications d'un nombre aléatoire r pour éviter un rejeu éventuel.

• (Obtention du ticket.) L'autorité C choisit la clé de session K au hasard et chiffre avec la clé K_A le message (K, r, L, B) où L représente l'intervalle de temps pendant lequel le ticket sera valide. L est composé de la date et de l'heure courantes, et d'un délai d'expiration. D'autre part, C chiffre avec la clé K_B le message (K, L, A), appelé le ticket. Les deux messages chiffrés sont envoyés à A.

• (Transmission du ticket.) Le client A déchiffre le premier message, qui lui révèle la valeur de K et il vérifie que la valeur de r reçue est identique à l'originale. Il transmet à B le ticket chiffré (tel qu'il l'a reçu) accompagné d'un message d'identification, chiffré avec K, comprenant sa propre identité et un nouvel horodatage h.

• (Identification de A.) Le serveur B déchiffre le ticket qui lui révèle la clé de session K. Il utilise cette clé pour déchiffrer le message d'identification. Il vérifie que l'identité de A contenue dans le ticket correspond à celle contenue dans le message d'identification, et que l'horodatage h et l'indication de sa propre horloge appartiennent à l'intervalle de validité L du ticket. En option pour permettre à A d'identifier B, celui-ci transmet à A la valeur h chiffrée à l'aide de K.

• (Identification optionnelle de B.) Dans ce cas, A déchiffre la valeur reçue et vérifie qu'elle correspond bien à h.

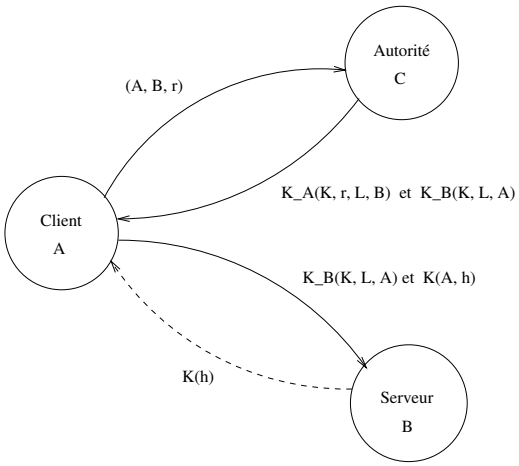

Figure 9. Protocole Kerberos

Justification

Bien que ce protocole ne soit pas formellement prouvé, on peut donner quelques indications pour justifier qu'il remplit bien les objectifs annoncés.

La vérification par B que l'identité de A contenue dans le ticket coïncide avec celle contenue dans le message d'identification lui permet d'identifier A. En effet, la version contenue dans le ticket est signée par C, et l'autre version est signée à l'aide de la clé de session, que seul A a été en mesure d'obtenir, en déchiffrant le message signé par C avec K_A.

Lorsque la partie optionnelle du protocole est effectuée, la dernière phase permet à A de s'assurer que la clé K en possession de B est correcte, et donc identifie B auprès de A puisque la connaissance de K_B a été nécessaire à B pour qu'il obtienne K.

Enfin, personne ne peut se substituer à l'autorité C car elle est la seule à connaître à la fois K_A et K_B. Le nombre aléatoire r empêche l'impersonnification de C par rejeu.

Remarques

Pour que le protocole fonctionne, il est nécessaire que les horloges des différents acteurs soient (à peu près) synchrones.

La clé de session K est générée par C. Une option du protocole non décrite ici permet de générer une seconde clé K' partagée par A et B mais non générée par C.

Tant que le ticket obtenu par A est en cours de validité, celui-ci peut l'utiliser d'autres fois pour de nouvelles identifications auprès de B sans faire intervenir C. Il lui suffit de retransmettre à B le ticket accompagné d'un nouveau message d'identification (avec un horodatage renouvelé). L'usage de l'option générant la clé K' permet alors d'utiliser des clés différentes à chaque fois.

Chapitre 14

Primalité

Les nombres premiers ayant de multiples propriétés utiles en cryptographie, il est important de savoir les reconnaître. Ce problème qui semble facile lorsqu'on s'intéresse seulement à des nombres de petite taille, devient plus délicat dans le domaine des nombres ayant 100 à 200 chiffres décimaux, important en cryptographie. Les livres [63], [111], et [112], bien que déjà anciens restent de bonnes références sur ce sujet.

Méthode naïve des divisions

La première méthode qui vient à l'esprit pour déterminer la nature d'un entier n est la méthode naïve des divisions (*trial division*). Elle consiste à tenter de diviser n successivement par 2, par 3, par 5..., et par tous les nombres impairs inférieurs à $\sqrt{n}$, en s'arrêtant dès que l'une de ces divisions mène à un reste nul. Dans le cas où n est premier, l'algorithme ne se termine que lorsque le diviseur atteint la borne $\sqrt{n}$, et requiert donc un nombre d'opérations élémentaires en $O\big(\sqrt{n}\ln(n)^2\big)$. Cela est prohibitif pour des nombres premiers de taille même modeste. Toutefois, cette méthode permet de déceler rapidement beaucoup de nombres composés. Précisément, la proportion de nombres non éliminés lorsqu'on tente les divisions jusqu'à une borne B est, d'après le théorème de Mertens,

$$\prod_{p<B}(1-\frac{1}{p}) \sim \frac{e^{-\gamma}}{\ln B} \qquad \text{où } \gamma \simeq 0.577 \text{ est la constante d'Euler.}$$

En pratique, face à un nombre de nature inconnue, on essaye d'abord la méthode naïve jusqu'à une borne raisonnable (par exemple 10^5). Si le nombre résiste à cette épreuve, on peut le suspecter d'être premier. L'étape suivante est de confirmer (ou infirmer) ces soupçons.

1. Tests de primalité

Les nombres premiers ont des tas de propriétés. Un test de primalité consiste à tester l'une de ces propriétés. Si elle n'est pas vérifiée alors le nombre considéré est nécessairement composé. Si elle est vérifiée alors on n'a toujours pas de certitude sur la nature de l'entier considéré, mais on est un peu plus persuadé qu'il est premier. Pour cette raison, on utilise parfois le terme plus approprié de *test de non-primalité* pour ce type de méthode.

Nombres pseudo-premiers

La première de ces propriétés est le petit théorème de Fermat 5.1.20. Si ce théorème est vrai pour les nombres premiers, il est souvent faux pour les nombres composés. Malheureusement, souvent n'est pas toujours.

1.1. — **Définition.** Soit $b \in \mathbb{N}^*$. Un entier $n \in \mathbb{N}^*$ est dit *probablement-premier de base b* s'il vérifie la relation suivante (qui implique que n et b soient premiers entre eux) :

$$b^{n-1} \equiv 1 \pmod{n}.$$

Si de plus n est composé, il est dit *pseudo-premier de base b*.

Le nombre $341 = 11 \cdot 31$ est pseudo-premier de base 2, le nombre $91 = 7 \cdot 13$ est pseudo-premier de base 3, le nombre $105 = 3 \cdot 5 \cdot 7$ est pseudo-premier de base 13...

1.2. — Exercice. Lister (par exemple à l'aide de Maple) les nombres pseudo-premiers de base 2 inférieurs à 500. Comparer avec la liste des nombres premiers.

1.3. — Exercice. Soit $n = 1387$. Calculer 2^{n-1} modulo n. Le nombre n est-il premier ?

Il existe donc des nombres probablement-premiers mais non premiers. Sont-ils nombreux ? Le Théorème des Nombres Premiers donne une approximation du nombre $\pi(x)$ de nombres premiers inférieurs ou égaux à x ($x \in \mathbb{R}_+$) :

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

En comparaison, le nombre de pseudo-premiers de base b inférieurs à x est compris entre $\exp(\ln x)^{5/4}$ et $x\sqrt{l(x)}$ pour x assez grand, où l est la fonction définie sur $\mathbb{R}_+^*$ par

$$l(x) = \exp\left(\frac{-\ln x \ln\ln\ln x}{\ln\ln x}\right).$$

Pour donner un exemple concret, il y a seulement 264 239 nombres pseudo-premiers de base 2 inférieurs à 10^{13} contre $\pi(10^{13}) = 37\,607\,912\,018$ nombres premiers. Le test de pseudo-primalité est donc relativement fiable. De plus, on peut l'utiliser plusieurs fois pour un même nombre, avec des bases différentes.

Nombres de Carmichael

Cependant il existe des nombres composés qui mettent totalement en défaut ce test. Ce sont les *nombres de Carmichael*, dont le plus petit est 561, et qui ont la propriété d'être pseudo-premiers de base b pour tout entier b premier avec eux. Plus précisément :

1.4. — Définition. Un nombre n est dit de *Carmichael* s'il est composé et s'il est pseudo-premier de base b pour chaque b tel que $\mathrm{pgcd}(b, n) = 1$. Cela signifie que l'indicateur de Carmichael $\lambda(n)$ (définition 5.1.19) divise $n - 1$.

Il existe 246 683 nombres de Carmichael inférieurs à 10^{16} alors que $\pi(10^{16})$ vaut 279 238 341 033 925. Selon un résultat de Alford/Granville/Pomerance [9], le nombre de nombres de Carmichael inférieurs à x est compris entre $x^{2/7}$ et $xl(x)$ pour x assez grand. Il y a donc peu de nombres de Carmichael, mais il y en a. Le test de pseudo-primalité est donc insuffisant.

Nombres pseudo-premiers forts

1.5. — Définitions. Soit $b \in \mathbb{N}^*$. Un entier impair $n \in \mathbb{N}^*$ est dit *probablement-premier fort de base b* s'il vérifie l'une des conditions suivantes, en posant $n - 1 = 2^k q$ avec q impair,

$$b^q \equiv 1 \pmod{n}$$

ou bien

$$\text{il existe un entier } i \text{ tel que } 0 \leqslant i < k \text{ et } b^{2^i q} \equiv -1 \text{ modulo } n.$$

Un entier composé probablement-premier fort de base b est dit *pseudo-premier fort de base b*.

En vertu du théorème 5.1.22, tout nombre premier p est probablement-premier de base b, pourvu que $p \nmid b$. D'autre part, tout pseudo-premier fort de base b est pseudo-premier de base b.

On ne connaît pas d'encadrement du nombre de pseudo-premiers forts de base b inférieurs à x autre que celui s'appliquant aux pseudo-premiers en général. Expérimentalement, il y a pourtant beaucoup moins de pseudo-premiers forts que de pseudo-premiers. Par exemple, il y a 58 897 pseudo-premiers forts de base 2 inférieurs à 10^{13}. De plus, le résultat suivant fait que la notion de pseudo-primalité forte représente un progrès majeur par rapport à celle de pseudo-primalité simple.

Pour n entier impair supérieur à 2, notons

$$B(n) = \{b \in \mathbb{Z}_n \mid n \text{ est probablement-premier fort de base } b\}.$$

1.6. — Théorème (Rabin [110], Monier [87]). *Si n est un entier composé distinct de 9 alors on a l'inégalité*

$$|B(n)| \leqslant \varphi(n)/4$$

où φ désigne la fonction indicatrice d'Euler. □

Le test de Rabin & Miller consiste à vérifier la pseudo-primalité forte pour un nombre k fixé (e.g. $k = 10$ ou 20) de bases prises au hasard. Son coût est en $O\big(\ln(n)^3\big)$.

Entrée : un entier $n > 1$ impair
Sortie : une réponse booléenne : "composé" ou "probablement premier"

$i := 20$
Tant que $i > 0$ faire
    Choisir au hasard un entier b dans $[2, n-1]$
    Si n n'est pas probablement-premier fort de base b alors
        retourner "composé" et arrêter l'algorithme
    Fin si
    $i := i - 1$
Fin tant que
Retourner "probablement premier"

Alors que le test de Fermat (où on utilise "probablement-premier de base b" au lieu de "probablement-premier fort de base b") souffre de l'existence des nombres de Carmichael, le théorème de Rabin-Monier montre que ce problème est résolu par l'utilisation du test de Rabin & Miller. De plus, lorsqu'on utilise k bases, la probabilité de ne pas déceler qu'un nombre est composé est inférieure à 4^{-k}.

Pseudo-premiers de Lucas

On peut aussi penser à utiliser les théorèmes 6.5.1 et 6.5.2. On obtient alors le test de Lucas et le test de Lucas fort. Soient donc P, Q deux entiers tels que $D = P^2 - 4Q$ ne soit pas un carré parfait.

1.7. — Définitions. On appelle *probablement-premier de Lucas de paramètres P et Q* tout entier $n > 1$, tel que $\operatorname{pgcd}(n, 2Q) = 1$ et

$$n \mid U_{n-\varepsilon}$$

où l'on désigne par ε le symbole de Jacobi (D/n). Si de plus n est composé, il est dit *pseudo-premier de Lucas de paramètres P et Q*.

1.8. — Définitions. On appelle *probablement-premier fort de Lucas de paramètres P et Q* tout entier $n > 1$ tel que $\operatorname{pgcd}(n, 2QD) = 1$ et

$$n \mid U_q \qquad \text{ou} \qquad \text{il existe un entier } i \text{ tel que } 0 \leqslant i < k \text{ et } n \mid V_{2^i q}$$

où l'on a posé $\varepsilon = (D/n)$ et $n - \varepsilon = 2^k q$ avec q impair. Si de plus n est composé, il est dit *pseudo-premier fort de Lucas pour les paramètres P et Q*.

1.9. — Exemples. Pour les paramètres $(P, Q, D) = (1, -1, 5)$, les entiers 323 et 377 sont pseudo-premiers de Lucas (avec symbole de Jacobi -1) mais pas pseudo-premiers forts de Lucas. Pour les même paramètres, les entiers 4181 (avec symbole de Jacobi $+1$) et 5777 (avec symbole de Jacobi -1) sont pseudo-premiers forts de Lucas.

Pour D entier et n tel que $\operatorname{pgcd}(n, 2D) = 1$, notons

$$C_D(n) = \left\{ (P, Q) \;\middle|\; \begin{array}{c} 0 \leqslant P, Q < n, \qquad P^2 - 4Q \equiv D \text{ modulo } n, \\ n \text{ est probablement-premier fort de Lucas pour } P \text{ et } Q \end{array} \right\}.$$

On a alors un résultat analogue au théorème de Rabin-Monier 1.6 :

1.10. — Théorème. *Soient D un entier et n un nombre composé distinct de 9 tel que* $\operatorname{pgcd}(n, 2D) = 1$ *alors on a l'inégalité*

$$|C_D(n)| \leqslant \frac{4}{15}n.$$

sauf peut-être si n est un produit de deux nombres premiers jumelés

$$n = (2^{k_1}q_1 - 1)(2^{k_1}q_1 + 1)$$

avec q_1 impair, $(2^{k_1}q_1 - 1/n) = -1$ et $(2^{k_1}q_1 + 1/n) = 1$. De plus, même dans ce cas exceptionnel, on a toujours

$$|C_D(n)| \leqslant n/2.$$

Démonstration — Voir [11]. □

Dans [15], on étudie la distribution des pseudo-premiers de Lucas et on en déduit plusieurs tests probabilistes de primalité analogues aux tests de Fermat et de Rabin. On y montre aussi qu'un algorithme testant à la fois des propriétés de pseudo-primalité au sens classique et au sens de Lucas constitue un test probabiliste de primalité particulièrement sûr. Plus précisément, Baillie et Wagstaff soulignent le fait que les nombres n qui sont simultanément pseudo-premiers dans une ou plusieurs bases données et pseudo-premiers de Lucas pour des paramètres P et Q donnés avec $(D/n) = -1$ sont expérimentalement particulièrement rares. Par exemple, on ne sait toujours pas s'il existe des entiers congrus à ± 2 modulo 5 (pour que $(5/n) = -1$) qui soient à la fois pseudo-premier de base 2 et pseudo-premiers de Lucas pour les paramètres $P = 1$, $Q = -1$ (donc $D = 5$).

2. Preuves de Primalité

Les tests de primalité permettent de déterminer la nature de grands entiers avec une probabilité d'erreur négligeable en pratique et aussi faible que voulue. Toutefois, lorsque ils indiquent qu'un entier est premier, ils n'apportent pas la preuve mathématique de ce fait. Si on désire des preuves, il faut utiliser des algorithmes beaucoup plus lourds, que nous allons introduire ici. Toutefois, les exercices suivants montrent que dans certains cas particuliers, les tests de primalité peuvent constituer une preuve.

2.1. — Exercice (critère de Pépin). • Soit n un entier de la forme $2^m + 1$ ($n \in \mathbb{N}^*$). Montrer que si m n'est pas une puissance de 2 alors n est composé.

• On désigne par F_n le n-ième nombre de Fermat : $F_n = 2^{2^n} + 1$, pour $n \in \mathbb{N}$. Déterminer le symbole de Jacobi $(3/F_n)$.

• Pour $n > 0$, montrer que F_n est premier si et seulement si $3^{(F_n-1)/2} \equiv -1$ modulo F_n.

2.2. — Exercice (test de Lucas-Lehmer). On désigne par M_n le n-ième nombre de Mersenne : $M_n = 2^n - 1$. • Montrer que si M_n est premier alors n aussi.

• On fixe un nombre premier p. Soit (S_k) la suite définie par $S_0 = 4$ et $S_{k+1} = S_k^2 - 2$ pour $k \geqslant 1$. Montrer que $S_k = V_{2^k}(4, 1)$ (suite de Lucas).

• Soit $\alpha = 2 + \sqrt{3}$. On désigne par q un facteur premier de M_p. Montrer que, si $q \mid S_{p-2}$, alors α est d'ordre $M_p + 1$ dans le groupe $\left(\mathbb{Z}[\sqrt{3}]/(q)\right)^*$.

• En déduire que, si $M_p \mid S_{p-2}$, alors M_p est premier. (Supposer que M_p possède un facteur premier $q < \sqrt{M_p}$ et en déduire une contradiction.)

• Montrer que $(3/M_n) = -1$, pour $n \geqslant 2$.

• On suppose désormais que M_p est premier. Montrer que l'anneau $A = \mathbb{Z}[\sqrt{3}]/(M_p)$ est isomorphe au corps $\mathbb{F}_{M_p^2}$.

• On note

$$A^\wedge = \{a + b\sqrt{3} + (M_p) \in A \mid a, b \in \mathbb{Z}, a^2 - 3b^2 \equiv 1 \pmod{M_p}\}.$$

Montrer que c'est un groupe cyclique d'ordre $M_p + 1$.

• Déterminer le symbole de Legendre $(2/M_p)$. En déduire que l'équation $1 + 6b^2 \equiv 2$ modulo M_p n'a pas de solution dans $\mathbb{Z}$.

• En déduire que α n'est pas un carré dans le groupe $A^\wedge$.
• En déduire que M_p est premier (avec p premier) si et seulement si $M_p \mid S_{p-2}$.

On a donc une méthode rapide pour déterminer si un nombre de Mersenne est premier. Le plus grand nombre premier connu est d'ailleurs $M_{13466917}$, un nombre de 4053946 chiffres décimaux.

Certificats de Pratt

Soit n un grand entier. Il est souvent difficile de savoir si n est premier ou composé. Toutefois, dans le cas où n est composé, il suffit d'exhiber un diviseur d pour convaincre quiconque de la nature de n, puisque la vérification que d divise n est aisée. Le coût de cette vérification étant polynomial, la non-primalité est un problème de décision de classe **NP**. Le statut du problème de la primalité est moins évident, mais analogue comme nous allons le montrer.

2.3. — **Théorème.** *Soit $n \geqslant 2$ entier. Alors n est premier si et seulement si le groupe $(\mathbb{Z}/n\mathbb{Z})^*$ contient un élément d'ordre $n-1$.*

Démonstration — Facile. □

2.4. — **Corollaire.** *Soit $n \geqslant 2$ un entier. Alors n est premier si et seulement si il existe un entier g tel que*

$$g^{n-1} \equiv 1 \pmod{n} \qquad \text{et} \qquad g^{(n-1)/q} \not\equiv 1 \pmod{n} \quad \text{pour tout diviseur premier } q \text{ de } n-1$$

2.5. — **Exercice.** Construire une preuve de la primalité de 79. La seule connaissance admise étant que 2 est premier.

La donnée des paramètres de 2.4 constitue donc une preuve de primalité aisément vérifiable, sous réserve que la factorisation de $n-1$ donnée soit complète, i.e. que les facteurs indiqués soient premiers. Ainsi, pour achever cette preuve, il suffit d'appliquer récursivement ce principe, jusqu'à ce que les facteurs impliqués soient suffisamment petits pour que leur primalité soit évidente. C'est le principe des *certificats de Pratt* [109]. L'utilisation de la récursivité fait que la preuve de primalité d'un entier repose sur la preuve de primalité de plusieurs entiers plus petits. Le fait intéressant est que le nombre total d'opérations nécessaires pour vérifier l'ensemble de la preuve est polynomial, comme le montrent les résultats suivants.

La primalité est de classe **NP**

Supposons n premier. Lorsque l'entier g est indiqué, ainsi que la factorisation de $n-1$, nous allons évaluer le temps de calcul nécessaire vérifier que les conditions du corollaire 2.4 sont respectées.

2.6. — **Lemme.** *Soient p un nombre premier, $q_1, \ldots, q_k$ les diviseurs premiers de $p-1$ et g un entier vérifiant les conditions du corollaire 2.4. Les entiers g et q_i étant indiqués (ainsi que leur exposant dans la factorisation de $p-1$), on désigne par $C(p)$ le nombre d'opérations nécessaires pour contrôler que les conditions du corollaire sont bien vérifiées.*

$$C(p) = C(q_1) + \cdots + C(q_k) + B(\ln^4 p)$$

où B est une constante.

Démonstration — Le coût de chaque exponentiation est cubique en $\ln(p)$ et il y en a $k+1$. Mais $k \leqslant \log_2(p)$, d'où le terme $B(\ln^4 p)$. La vérification de $p-1 = q_1 \cdots q_k$ est de coût quadratique, donc négligeable. Il reste le coût des vérifications de la primalité des q_i, exprimé par les autres termes. □

2.7. — **Lemme.** *Soient $\alpha_1, \ldots, \alpha_k \geqslant 1$ des réels ($k \geqslant 2$). Alors on a*

$$\Big(\sum_{i=1}^{k} \alpha_i\Big)^5 \geqslant \sum_{i=1}^{k} \alpha_i^5 + \Big(\sum_{i=1}^{5} \alpha_i\Big)^4.$$

DÉMONSTRATION — On peut supposer $\alpha_1 \leqslant \alpha_i$ pour $1 \leqslant i \leqslant k$. On a alors

$$\Big(\sum_{i=1}^{k}\alpha_i\Big)^5 = \alpha_1\Big(\sum_{i=1}^{k}\alpha_i\Big)^4 + \Big(\sum_{i=2}^{k}\alpha_i\Big)\Big(\alpha_1+\Big(\sum_{i=2}^{k}\alpha_i\Big)\Big)^4 \geqslant \Big(\sum_{i=1}^{k}\alpha_i\Big)^4 + \Big(\sum_{i=2}^{k}\alpha_i\Big)\alpha_1^4 + \Big(\sum_{i=2}^{k}\alpha_i\Big)^5$$
$$\geqslant \Big(\sum_{i=1}^{k}\alpha_i\Big)^4 + \alpha_1^5 + \Big(\sum_{i=2}^{k}\alpha_i\Big)^5 \geqslant \Big(\sum_{i=1}^{k}\alpha_i\Big)^4 + \sum_{i=1}^{k}\alpha_i^5.$$

□

2.8. — **Lemme.** *On reprend les notations du lemme 2.6, avec $p \geqslant 5$. Supposons qu'il existe une constante $A \geqslant 2B$ telle que $C(q) \leqslant A(\ln q)^5$ pour tout premier $q < p$. Alors $C(p) \leqslant A(\ln p)^5$.*

DÉMONSTRATION — On a

$$C(p) = \sum_{i=1}^{k} C(q_i) + B\ln^4(p) \leqslant \sum_{i=1}^{k} A\ln^5(q_i) + \frac{A}{2}\ln^4(p)$$
$$\leqslant A\Big(\sum_{i=1}^{k}\ln^5(q_i) + \ln^4(p-1)\Big) \qquad (\text{car } p \geqslant 5)$$
$$= A\Big(\sum_{i=1}^{k}\ln^5(q_i) + \Big(\sum_{i=1}^{k}\ln(q_i)\Big)^4\Big) \leqslant A\Big(\sum_{i=1}^{k}\ln(q_i)\Big)^5 \leqslant A\ln^5(p).$$

□

2.9. — **Théorème.** *La primalité est un problème de décision de classe* **NP**.

DÉMONSTRATION — Soit B la constante intervenant dans le lemme 2.6 et soit $A \geqslant 2B$ tel que $C(2) \leqslant A\ln^5(2)$ et $C(3) \leqslant A\ln^5(3)$. D'après 2.8, on a $C(p) \leqslant A\ln^5(p)$ pour tout p premier. D'où le résultat. □

Théorème de Pocklington

La généralisation suivante de 2.3 est très intéressante puisqu'elle donne une information même lorsqu'on ne connaît qu'une factorisation partielle de $n-1$. C'est très utile dans la pratique parce qu'une factorisation partielle d'un entier est souvent bien plus facile à obtenir qu'une factorisation complète.

2.10. — **Théorème (Pocklington).** *Soient $n > 1$ un entier et $s > 0$ un diviseur de $n-1$ tels qu'il existe un entier b vérifiant*

$$b^{n-1} \equiv 1 \bmod n \qquad \textit{mais} \quad b^{(n-1)/q} - 1 \textit{ étranger à } n \textit{ pour tout diviseur premier } q \textit{ de } s.$$

Alors, tout diviseur premier de n est congru à 1 modulo s. En conséquence, si $s > \sqrt{n}-1$ alors n est premier.

DÉMONSTRATION — Soit $t = (n-1)/s$. Les hypothèses expriment que b^t est d'ordre s modulo chaque diviseur premier p de n. Donc $s \mid p-1$, d'où la première affirmation. Si $s > \sqrt{n}-1$ alors tout diviseur premier p de n vérifie $p \geqslant s+1 > \sqrt{n}$. Ceci est impossible si n est composé. □

Théorème de Morrison

Les suites de Lucas donnent un critère semblable utilisant une factorisation partielle de $n+1$:

2.11. — **Théorème (Morrison).** *Soient $n \in \mathbb{N}^*$ et $s > 0$ un diviseur de $n+1$ tels que* $\operatorname{pgcd}(s,(n+1)/s) = 1$ *et, pour des paramètres (P,Q) de Lucas,*

$$U_{n+1} \equiv 0 \bmod n \qquad \textit{mais} \quad U_{(n+1)/q} \textit{ étranger à } n \textit{ pour tout diviseur premier } q \textit{ de } s.$$

Alors, tout diviseur premier de n est congru à ± 1 modulo s.

DÉMONSTRATION — Voir [89] ou [133]. □

Ces résultats et d'autres (en particulier un faisant intervenir des factorisations partielles de $n-1$ **et** $n+1$) sont discutés dans le livre [28].

Utilisation des courbes elliptiques

Ainsi, si l'on sait (partiellement) factoriser $n-1$, on peut prouver la primalité de n. De même si l'on sait (partiellement) factoriser $n+1$. Par factorisation paresseuse de $n-1$ ou $n+1$ se restreignant aux facteurs premiers inférieurs à 30030, on peut prouver la primalité de n'importe quel nombre premier inférieur à 10^{80}. Mais pour des nombres plus grands, il se peut que $n-1$ et $n+1$ soient tous les deux difficiles à factoriser. Les courbes elliptiques permettent alors d'étendre largement le choix jusque-là restreint à $n-1$ et $n+1$.

Une difficulté théorique apparaît dans l'utilisation des courbes elliptiques en factorisation, puisqu'on a besoin de courbes elliptiques définies sur un anneau $\mathbb{Z}/n\mathbb{Z}$ avec n non premier. En pratique, on peut se contenter de calculer sur une telle courbe "comme si n était premier", en contrôlant que toutes les opérations modulo n effectuées ont un sens. Si l'une d'elle n'a pas de sens (division par un élément non-inversible), c'est une aubaine pour qui veut factoriser.

2.12. — Théorème. *Soit $n>1$ un entier étranger à 6. Soient E une courbe elliptique sur $\mathbb{Z}/n\mathbb{Z}$ et m, s deux entiers tels que $s \mid m$. Supposons que E possède un point P tel que*

$$mP = O \qquad \text{mais la coordonnée } z \text{ de } (m/q)P \text{ est étrangère à } n \text{ pour tout diviseur premier } q \text{ de } s.$$

Alors, pour tout diviseur premier p de n, le nombre de points de E sur $\mathbb{Z}/p\mathbb{Z}$ est un multiple de s. En conséquence, si $s > (\sqrt[4]{n}+1)^2$ alors n est premier.

Démonstration — Soit $t = m/s$. Les hypothèses expriment que tP est d'ordre s sur la courbe modulo p, pour chaque diviseur premier p de n. Donc s divise l'ordre m_p de cette courbe. Si $s > (\sqrt[4]{n}+1)^2$ alors

$$p+1+2\sqrt{p} \geqslant m_p \geqslant s > \sqrt{n}+1+2\sqrt[4]{n}.$$

(La première inégalité est due au théorème de Hasse 6.6.15.) On obtient $p+2\sqrt{p} > \sqrt{n}+2\sqrt[4]{n}$ donc $p > \sqrt{n}$. Mais cette inégalité ne peut être vérifiée par tout facteur premier de n si celui-ci est composé. □

C'est ce critère qui est utilisé dans l'algorithme d'Atkin/Morain [12]. Les courbes y sont choisies de telle manière que l'ordre m se décompose en produit de nombres premiers

$$m = \Big(\prod_{i=1}^{r} q_i\Big)t \tag{1}$$

où les q_i sont de petits facteurs premiers (donc faciles à trouver et à prouver premiers) et où le dernier facteur t est supérieur à $(\sqrt[4]{n}+1)^2$ et probablement premier. Si les conditions du critère sont réunies, alors elles prouvent que n est premier, sous réserve que t soit aussi prouvé premier. On construit ainsi une preuve récursive de la primalité de n.

Pour donner un ordre d'idée des performances pratiques de cet algorithme, citons le nombre $10^{5019}+43157099231631693$ dont le certificat de primalité a été établi en septembre 2001 en 13 semaines de calcul sur un PC à 1300 MHz.

Citons aussi l'algorithme d'Adleman/Pomerance/Rumely/Cohen/Lenstra [4] et [32] qui est antérieur à l'algorithme d'Atkin/Morain et n'utilise pas de courbes elliptiques mais des sommes de Gauss. Toutefois, cet algorithme ne construit pas de preuve de primalité rapidement vérifiable. Enfin, Adleman et Huang [3] ont décrit un algorithme (impraticable, semble-t-il) montrant que le problème de la primalité est dans la classe **RP**.

3. Un algorithme déterministe polynomial

Bien que prouvée dès 1976 dans [86] sous l'hypothèse de Riemann étendue (ERH), l'appartenance du problème de la primalité à la classe **P** est restée incomplètement résolue jusqu'en 2002. A cette date, les auteurs de [6] répondent positivement à la question à l'aide d'un algorithme assez simple. Cet algorithme, bien que de complexité polynomiale, a un coût prohibitif pour des nombres de taille raisonnable. Ceci fait qu'il n'a aucun intérêt pratique. Toutefois, ce résultat est d'importance théorique capitale. Cet algorithme est basé sur le résultat suivant.

Le théorème principal

3.1. — Théorème. *Soit $N \in \mathbb{N}^*$. Soient r et q deux premiers impair avec $q \mid r-1$ et*

$$N^{(r-1)/q} \not\equiv 0, 1 \pmod{r}. \tag{2}$$

Soit $s \in \mathbb{N}^*$ tel que

$$C^s_{s+q-1} \geqslant N^{2\lfloor\sqrt{r}\rfloor} \tag{3}$$

et tel que N n'ait pas de diviseur dans l'intervalle $[2, s]$. Si l'identité

$$(X-a)^N \equiv X^N - a \pmod{(X^r-1, N)}$$

est vérifiée pour tout entier a contenu dans $[1, s]$, alors N est la puissance d'un nombre premier.

3.2. — Lemme. *Les conditions*

$$q \geqslant 4\lceil\sqrt{r}\log_2 N\rceil \quad \textit{et} \quad s = 2\lceil\sqrt{r}\log_2 N\rceil \tag{4}$$

impliquent la condition (3).

Démonstration — Si les conditions (4) sont satisfaites alors

$$C^s_{s+q-1} = \frac{(s+q-1)\cdots(q+1)q}{s\cdots 2\cdot 1} \geqslant \left(\frac{q}{s}\right)^s \geqslant 2^s \geqslant 2^{2\sqrt{r}\log_2(N)} = N^{2\sqrt{r}} \geqslant N^{2\lfloor\sqrt{r}\rfloor}.$$

Et voilà. □

Un peu de combinatoire

3.3. — Lemme. *Soient $s, q \in \mathbb{N}$. Le nombre de s-uplets d'entiers $(e_1, \ldots, e_s) \in \mathbb{N}^s$ tels que $e_1 + \cdots + e_s \leqslant q$ est C^s_{s+q}. C'est aussi le nombre de $(s+1)$-uplets $(e_0, \ldots, e_s) \in \mathbb{N}^{s+1}$ tels que $e_0 + \cdots + e_{s-1} = q$.*

Démonstration — Il est clair que les deux nombres annoncés sont égaux. Notons-les $A_{s,q}$. On voit que $A_{0,q} = 1 = C^0_q$ pour tout q. L'identité classique

$$C^k_n = \sum_{i=k-1}^{n-1} C^{k-1}_i \qquad \text{pour } n, k \in \mathbb{N}$$

donne ici

$$C^s_{s+q} = \sum_{i=s-1}^{s+q-1} C^{s-1}_i = \sum_{i=0}^{q} C^{s-1}_{s-1+i}$$

Si l'on suppose la propriété vraie pour $s-1$, on obtient $C^s_{s+q} = \sum_{i=0}^q A_{s-1,i}$. Mais il est aussi clair que

$$\sum_{i=0}^{q} A_{s-1,i} = \sum_{i=0}^{q} \#\{s\text{-uplets de somme } i\} = \#\{s\text{-uplets de somme} \leqslant q\} = A^q_s.$$

On obtient donc le résultat par récurrence sur s. □

Démonstration du théorème 3.1

Il existe un diviseur premier p de N tel que

$$p^{(r-1)/q} \not\equiv 0, 1 \pmod{r}.$$

Soit d l'ordre de p modulo r. D'après l'équation précédente, c'est un multiple de q, donc $d \geqslant q$. Le polynôme cyclotomique $X^{r-1} + \cdots + X + 1$ se factorise sur $\mathbb{F}_p$ en produit d'irréductibles de degré d (voir par exemple le théorème 13.2 dans [53]). Soit h l'un d'entre eux et désignons par K une extension du corps $\mathbb{F}_p$ définie par le polynôme h.

Soit G le sous-groupe (cyclique) de K^* engendré par les $X - a$ pour $1 \leqslant a \leqslant s$. Comme les $X - a$, pour $1 \leqslant a \leqslant s$, sont des irréductibles distincts dans $\mathbb{F}_p[X]$ et comme $\deg h \geqslant q$, les produits $(X-1)^{e_1} \cdots (X-s)^{e_s}$ avec $e_1 + \cdots + e_s < q$ sont tous distincts modulo h. Donc l'ordre de G est au moins $C^s_{s+q-1} \geqslant N^{2\lfloor\sqrt{r}\rfloor}$ d'après le lemme 3.3.

Considérons dans $\mathbb{F}_p[X]$ un polynôme g qui, modulo h, engendre G. Par hypothèse, pour $1 \leqslant a \leqslant s$, on a l'identité $(X - a)^N \equiv X^N - a$ modulo $(X^r - 1, p)$. On a aussi $(X - a)^p \equiv X^p - a$ modulo $(X^r - 1, p)$. Donc, pour tout couple (i, j) d'entiers on a, pour $1 \leqslant a \leqslant s$,

$$(X - a)^{N^i p^j} \equiv (X^{N^i p^j} - a) \pmod{X^r - 1, p}$$

et donc

$$g(X)^{N^i p^j} \equiv g(X^{N^i p^j}) \pmod{X^r - 1, p}.$$

Parmi les couples (i, j) tels que $0 \leqslant i, j \leqslant \sqrt{r}$, il y en a deux tels que $N^{i_1} p^{j_1} \equiv N^{i_2} p^{j_2}$ modulo r (car il y a plus de r tels couples au total). On a alors

$$g(X^{N^{i_1} p^{j_1}}) \equiv g(X^{N^{i_2} p^{j_2}}) \pmod{h}$$

et donc

$$g(X)^{N^{i_1} p^{j_1}} \equiv g(X)^{N^{i_2} p^{j_2}} \pmod{h}.$$

Cela implique $N^{i_1} p^{j_1} \equiv N^{i_2} p^{j_2}$ modulo l'ordre de G.

Mais $0 \leqslant N^{i_1} p^{j_1}, N^{i_2} p^{j_2} \leqslant N^{2\lfloor\sqrt{r}\rfloor}$ donc $N^{i_1} p^{j_1} = N^{i_2} p^{j_2}$. Comme les couples (i_1, j_1) et (i_2, j_2) sont différents. Cela implique $N = p$. □

L'algorithme

D'après le théorème 3.1 dans le cas particulier du lemme 3.2, l'algorithme ci-dessous fournit une réponse exacte au problème de la primalité.

Entrée : un entier $N > 1$.
Sortie : la réponse à la question "N est-il premier ?".

Si N est de la forme a^b avec $b \geqslant 2$ répondre "non".
Pour $r = 3$ initialement et incrémenté par pas de 2 répéter
    Si $\text{pgcd}(r, N) > 1$ alors répondre "non".
    Si r n'est pas premier alors itérer.
    $q :=$ le plus grand facteur premier de $r - 1$.
    Si $q < 4\sqrt{r} \log_2 N$ alors itérer.
    Si $N^{(r-1)/q} \equiv 1$ modulo r alors itérer.
    Sortir de la boucle.
– – Ici, r est un premier vérifiant (2) et $s = 2\lceil\sqrt{r} \log_2 N\rceil$ vérifie (3).
Pour a de 1 à $\lceil 2\sqrt{r} \log_2 N\rceil$ répéter
    Si $a \mid N$ et $a \neq 1$ alors répondre "non".
    Si $(X - a)^N \not\equiv X^N - a$ modulo $(X^r - 1, N)$ alors répondre "non".
Répondre "oui".

Estimation du coût

Le test critique (en fin d'algorithme) fait intervenir des exponentiations dans l'anneau $\mathbb{Z}[X]/(X^r-1,N)$ où la représentation de chaque élément est de taille $r\log_2 N$ bits. Un multiplication dans cet anneau coûte $\mathcal{O}'(r\ln N)$ opérations binaires si l'on utilise la transformée de Fourier rapide (la notation $\mathcal{O}'(f)$ signifie ici $\mathcal{O}(f^{1+\varepsilon})$). Le test critique coûte alors $\mathcal{O}'(r\ln^2 N)$ opérations et la dernière boucle $\mathcal{O}'(r^{3/2}\ln^3 N)$ opérations.

Un théorème de Fouvry permet d'obtenir une borne en $\mathcal{O}(\ln^6 N)$ sur le nombre premier r fourni par la première boucle. Cela donne un coût en $\mathcal{O}'(\ln^{12} N)$ opérations pour la dernière boucle. Le reste de l'algorithme est de coût inférieur (même en utilisant des méthodes naïves pour la primalité de r et la factorisation de $r-1$). Cela montre un coût total polynomial en $\mathcal{O}'(\ln^{12} N)$ pour la primalité de N. Bien que polynomial, ce coût est trop élevé pour une application pratique. Mais ces travaux ne sont pas à l'abri d'éventuelles améliorations...

4. Génération aléatoire de clés cryptographiques

Pour construire des clés pour les principaux systèmes cryptographiques asymétriques, il faut savoir choisir au hasard des nombres premiers de taille voulue. Cette tâche peut s'avérer plus ou moins complexe suivant que les caractéristiques des nombres premiers recherchés (prouvés premiers ou seulement probablement premiers, prouvés résistants aux méthodes de factorisation $p\pm 1$ ou non, uniformément distribués ou non, etc). Le livre [84] est assez riche sur ce thème.

La méthode la plus simple consiste à générer des entiers (impairs) au hasard de taille voulue et à les filtrer par un bon test de primalité. Elle fournit des nombres premiers uniformément aléatoires mais est lente et utilise beaucoup d'aléa car nécessite souvent de générer beaucoup d'entiers avant d'en trouver un qui soit premier (en moyenne environ $256\ln(2)\simeq 175$ pour un premier de taille 512 bits).

Une variante consiste à générer un seul entier impair au hasard et à tester sa primalité, ainsi que celle de ses successeurs par pas de 2 jusqu'à obtenir un nombre premier. Cette méthode utilise peu d'aléa mais est biaisée puisque les termes de la suite des nombres premiers précédés par un écart important sont favorisés sur les autres.

Cette section décrit quelques méthodes pour générer des nombres premiers (ou des couples de nombres premiers formant une clé DSA) avec des contraintes spécifiques.

La méthode de Gordon

Elle génére des nombres premiers p non-prouvés mais *résistants* dans le sens où ils vérifient les trois conditions suivantes (ces conditions permettent de fabriquer des modules RSA qui ne soient pas fragiles vis-à-vis d'attaques classiques).

$p-1$ est divisible par un "grand" premier r,
$p+1$ est divisible par un "grand" premier s,
$r-1$ est divisible par un "grand" premier t.

Les deux premières conditions peuvent s'écrire $p\equiv 1$ modulo r et $p\equiv -1$ modulo s auxquelles on peut rajouter $p\equiv 1$ modulo 2. Le tout étant équivalent à $p\equiv u$ modulo $2rs$ avec $u=2(s^{-1} \bmod r)s-1$. Cela justifie l'algorithme suivant (la taille d'un entier est le nombre de rangs dans son écriture binaire) :

Entrée : un entier k
Sortie : un nombre (probablement) premier *résistant* et de taille k bits.

Choisir ϵ voisin de $\log_2 k$
Choisir t premier de taille voisine de $k/2-2\epsilon$
Choisir i de taille ϵ tel que $r=2it+1$ soit premier
Choisir s premier de taille voisine de $k/2-\epsilon$
$u \leftarrow 2(s^{-1} \bmod r)s-1$
Choisir p premier de taille k et de la forme $u+2vrs$ (avec v de taille proche de 2ϵ)
Retourner p

Algorithme 1. Méthode de Gordon

La méthode de Maurer

Elle génère des nombres premiers prouvés, de taille voulue, et avec une distribution proche de la distribution uniforme. Elle se présente sous la forme d'un algorithme qui produit un nombre premier p de la forme $p = 2FR + 1$ où F vérifie $2F > R$ et est une factorisation partielle de $p - 1$ générée récursivement. Cette factorisation partielle est utilisée pour prouver la primalité de p à l'aide du critère de Pocklington 2.10. Cette méthode est décrite dans [81].

Entrée : Deux bornes P_1 et P_2.
Sortie : Un nombre premier dans l'intervalle $[P_1, P_2]$.
Constantes : B, c de valeurs respectives conseillées 2^{20} et 1,2.

Si $P_2 \leqslant B$ alors $\quad --$ se contenter d'une méthode naïve
  Répéter
    Choisir p au hasard dans $[P_1, P_2]$
  Jusqu'à ce que p n'ait pas de facteur dans $[2, \sqrt{p}]$
  Retourner p et terminer
Fin (si)
$P \leftarrow \sqrt{(P_1 - 1)(P_2 - 1)}/2$ (valeur approchée)
$S \leftarrow (s_1, \ldots, s_r)$ (une liste de taille variable dont la génération est décrite ci-dessous)
$F \leftarrow 1$
Pour i de 1 à r faire
    $Q \leftarrow P^{s_i}$
    $q_i \leftarrow \text{Maurer}(Q/c, Qc)$
    $F \leftarrow q_i F$
Fin (Pour) $\quad --$ Ici, $2F$ est de l'ordre de $P^{s_1 + \cdots + s_r}$, donc $2F \geqslant \sqrt{P}$ si (5) est satisfaite
$I_1 \leftarrow \lceil (P_1 - 1)/2F \rceil$
$I_2 \leftarrow \lfloor (P_2 - 1)/2F \rfloor$
Répéter
    Choisir R au hasard dans $[I_1, I_2]$
    $p \leftarrow 2RF + 1$
    Si p a des petits facteurs alors itérer
    Si p n'est pas premier (utiliser le critère de Pocklington) alors itérer
    Retourner p et terminer
Fin (répéter)

Algorithme 2. Maurer(P_1, P_2)

Pour que la distribution soit bonne, le quotient P_2/P_1 ne doit pas être trop grand (2 est une valeur fréquente et très acceptable).

Les tailles relatives s_i sont des réels positifs décroissants et vérifiant la relation

$$0 < 1 - (s_1 + \cdots + s_r) < s_r. \tag{5}$$

Cette relation implique en particulier $s_1 + \cdots + s_r > 1/2$ ce qui permet d'utiliser le critère de Pocklington. Elles sont générées à l'aide de l'algorithme décrit ci-après

Notons $p_i(n)$ le i-ème plus grand facteur premier d'un entier n (lorsqu'il existe) et posons

$$\omega_i(x, y) = \#\{n | 1 \leqslant n \leqslant x, p_i(n) \leqslant x^y\}$$

Sortie : une liste décroissante $(s_1, \ldots, s_r)$ (de taille r variable).

$r \leftarrow 0$
$t \leftarrow 1$
Répéter
    $r \leftarrow r + 1$
    Choisir s_r au hasard dans $]0, t[$
    $t \leftarrow t - s_r$
    Trier les $s_1, \ldots, s_r$ par ordre décroissant
    Si $t < s_r$ alors sortir de la boucle
Fin (Répéter)
Retourner $(s_1, \ldots, s_r)$

Algorithme 3. Génération des tailles relatives $s_1, \ldots, s_r$

ainsi que $F_i(x) = \lim_{x \to \infty} \omega_i(x, y)/x$. Les fonctions F_i ont été étudiées dans [57] où il est montré en particulier que $F_1(x) = 1 + \ln(x)$ lorsque $1/2 \leqslant x \leqslant 1$. La procédure indiquée de génération de tailles relatives est conçue pour respecter les proportions exprimées par les F_i.

La méthode du NIST pour générer des clés DSA

Rappelons qu'il s'agit de deux nombres premiers p, q de tailles respectives $L = 512 + 64\ell$ (où $0 \leqslant \ell \leqslant 8$) et $M = 160$ bits, avec $q \mid p - 1$.

Cette méthode est décrite dans [93] et utilise la fonction SHA-1 (notée ici h) pour générer du pseudo-aléa à partir d'un germe s aléatoire. La donnée de ce germe permet de vérifier que la clé à été construite suivant les règles et limite le risque de production intentionnelle de clés faibles. Attention : la taille de ce germe détermine l'entropie de la clé construite.

La méthode utilise un test de primalité *robuste* défini par le NIST comme étant un test pour lequel la probabilité qu'un composé soit déclaré premier est inférieure à 2^{-80}.

Entrée : L'entier L et un germe s de longueur $g \geqslant 160$ bits.
Sortie : Une clé DSA p, q avec p de longueur L (ou "échec").

$q \leftarrow h(s) \oplus h(s + 1 \bmod 2^g)$
Forcer les bits extrêmes de q à 1 (pour que q soit impair et de longueur M)
Tester la primalité de q avec un test de primalité robuste
Si q n'est pas premier, retourner "échec" et terminer
$(n, r) \leftarrow$ quotient et reste de la division euclidienne de $L - 1$ par M
Pour i de 0 à 4095 faire
    Pour k de 0 à n faire $V_k \leftarrow h\big(s + 2 + k + i(n + 1) \bmod 2^g\big)$
    – – Construire l'entier p en concaténant L bits :
    $p \leftarrow 1 \,\|\, V_n \bmod 2^r \,\|\, V_{n-1} \,\|\, \ldots \,\|\, V_0$
    $p \leftarrow p - (p \bmod 2q) + 1$
    Si $p \geqslant 2^{L-1}$ alors
        Tester la primalité de p avec un test de primalité robuste
        Si p est déclaré premier alors retourner p, q et terminer
    Fin (Si)
Fin (Pour)
Retourner "échec"

Algorithme 4. Génération de clés DSA, méthode du NIST

Génération de clés DSA par GnuPG

Elle est en fait aussi utilisée pour générer des clés El Gamal, avec un sous-groupe d'ordre q. La longueur M de q est 160, 200 ou 240, selon la longueur L de p. Cette méthode est conçue pour que les facteurs premiers de $(p-1)/2$ soient tous de taille $\geqslant M$ (ou à peu près). Ces facteurs premiers sont connus au moment de la génération et il est donc en principe possible de certifier la primalité de p. La méthode permet à GnuPG de générer des clés DSA de taille 1024 bits en une (petite) poignée de secondes sur un ordinateur modeste. L'algorithme qui suit en est une description approximative.

Entrée : Les longueurs L et M.
Sortie : Deux premiers p, q de longueurs respectives L et M et tels que $q \mid p-1$.

Choisir un premier q de longueur M — avec une méthode naïve, car M est petit
$k \leftarrow \lfloor (L-M-1)/M \rfloor$ — nombre de facteurs premiers impairs de $p-1$
$m \leftarrow \lceil (L-M-1)/k \rceil$ — donc $m \geqslant M$
Choisir un entier n tel que C_n^k soit nettement plus grand que $\ln(2^L)$
Répéter
    Choisir n premiers $q_1, \ldots q_n$ de taille (voisine de) m
    Répéter jusqu'à épuisement des ensembles I possibles
        Choisir un (autre) ensemble I de k entiers distincts de 1 à n
        $p \leftarrow 2q \prod_I q_i + 1$
        Si p est de taille L alors
            Tester la primalité de p
            Si p est premier alors retourner (p, q) et terminer
        Fin Si
        Si "il s'avère difficile" de trouver des p de taille L exactement, alors
            Changer certains q_i et leur taille (± 1 bit).
        Fin Si
    Fin Répéter
Fin Répéter — échec provisoire, renouveler les q_i

Algorithme 5. Génération de clés DSA par GnuPG

Chapitre 15

Factorisation

Lorsqu'on dispose d'un grand entier que l'on sait composé (parce qu'on lui a appliqué un test de primalité, ou bien parce que c'est un module RSA par exemple), il peut être difficile (heureusement pour RSA) de le factoriser, c'est-à-dire déterminer explicitement sa décomposition en produit de facteurs premiers. Le but de ce chapitre est d'introduire les principales méthodes connues pour cette tâche.

Scinder un entier N veut dire en déterminer un facteur non-trivial (i.e. différent de ± 1 et $\pm N$). Il est clair que la tâche de la factorisation complète se ramène à scinder puisqu'il suffit, lorsqu'on a réussit à scinder N, de déterminer si les facteurs d et N/d obtenus sont premiers ou composés et, lorsque ils sont composés, de leur appliquer récursivement la méthode.

Petits facteurs

Le problème de scinder un entier est bien souvent trivial (un nombre sur deux est pair !). La première chose à faire face à un nombre de nature inconnue est donc d'effectuer une recherche exhaustive de petits facteurs. On tente donc la division par 2, puis par 3, 5, 7,... jusqu'à une borne fixée B. En principe, il suffirait de ne tenter les divisions que par les nombres premiers inférieurs à B. C'est possible si on dispose d'une table de ces nombres, mais c'est très coûteux en mémoire. Dans la pratique, on se contente d'éviter les divisions par les nombres pairs et les multiples de 3 (sauf 2 et 3 !), voire les multiples de 5 et même 7. Une alternative intéressante est de calculer le pgcd de N avec le produit pré-calculé de tous les nombres premiers inférieurs à B. C'est une façon très rapide de savoir si N a des petits facteurs.

Ce n'est que lorsque l'entier considéré à été débarrassé de ses petits facteurs que les méthodes suivantes prennent leur intérêt. De plus, si cet entier est de nature inconnue, c'est le moment d'employer un bon test de primalité, avant de perdre son temps en lançant de coûteuses méthodes de factorisation...

Entiers friables

0.1. — **Définitions.** Soit $B \in \mathbb{N}^*$. • Un entier $n \in \mathbb{N}^*$ est dit *B-friable* (*B-smooth*) si tout diviseur premier p de n vérifie $p \leqslant B$.

• Il est dit *fortement B-friable* si toute puissance p^r d'un nombre premier telle que $p^r \mid n$ vérifie $p^r \leqslant B$.

Un principe universel

La plupart des méthodes pour scinder un entier composé N passent par la détermination de deux entiers x, y vérifiant les congruences

$$x^2 \equiv y^2 \pmod{N} \qquad \text{mais} \qquad x \not\equiv \pm y \pmod{N}. \tag{1}$$

En effet, ces congruences signifient que $x - y$ et $x + y$ sont des diviseurs de zéro dans l'anneau $\mathbb{Z}/N\mathbb{Z}$, donc que $\operatorname{pgcd}(x - y, N)$ et $\operatorname{pgcd}(x + y, N)$ sont des facteurs non triviaux de N.

1. Les méthodes classiques

La méthode de Fermat

La méthode qui suit est une adaptation de celle que Fermat utilisait pour factoriser des entiers produits de deux facteurs proches l'un de l'autre. On suppose que $N = uv$ est composé impair et $u \leqslant v$. Alors, en posant $x = (u+v)/2$ et $y = (v-u)/2$, on a $N = x^2 - y^2$. L'idée de Fermat était de calculer $x^2 - N$ pour des valeurs successives de x en commençant par $x = \left\lceil \sqrt{N} \right\rceil$ (où $\lceil x \rceil$ désigne le plus petit entier supérieur ou égal à x) et

en incrémentant x à chaque étape. Par quelques astuces, Fermat arrivait sans trop de mal à reconnaître les $x^2 - N$ qui sont des carrés y^2. En particulier, un carré parfait se termine toujours par les chiffres

$$00, \quad a1, \quad a4, \quad 25, \quad b6, \quad \text{ou } a9 \qquad \text{avec } a \text{ chiffre pair et } b \text{ chiffre impair.}$$

On finit par obtenir ainsi les plus petits entiers positifs x, y vérifiant $N = x^2 - y^2$ et il suffit d'appliquer alors le principe (1) pour obtenir des facteurs non-triviaux de N. Puisque la recherche se fait en commençant par les petites valeurs de y, les deux facteurs obtenus sont les plus proches l'un de l'autre.

Une formulation moderne de la méthode de Fermat est la suivante.

Entrée : N entier composé impair.
Sortie : les deux facteurs $u \leqslant v$ les plus proches de $\sqrt{N}$.

$x := \left\lceil \sqrt{N} \right\rceil$
$z := x^2 - N$
Tant que z n'est pas un carré parfait répéter
    $z := z + 2x + 1$
    $x := x + 1$
Fin tant que
$y := \sqrt{z}$
Retourner $(x - y, x + y)$

Soient u le plus grand diviseur de N qui soit inférieur à $\sqrt{N}$ et v le plus petit qui soit supérieur à $\sqrt{N}$. Le nombre d'itérations de la boucle effectuées dans cet algorithme est

$$\frac{u+v}{2} - \left\lceil \sqrt{N} \right\rceil \simeq \frac{u+v}{2} - \sqrt{uv} = \frac{1}{2}(\sqrt{u} - \sqrt{v})^2 = \frac{(u-v)^2}{2(\sqrt{u} + \sqrt{v})^2}.$$

Lorsque u et v sont proches l'un de l'autre, et donc de $\sqrt{N}$, le nombre de tours de boucles effectués est proche de $(u-v)^2/8\sqrt{N}$. La méthode de Fermat reste donc efficace tant que la différence $u - v$ reste de l'ordre de $\sqrt[4]{N}$.

Voici une version optimisée "à la Knuth" où, pour des raisons d'efficacité, on n'utilise pas x et y mais $x' = 2x + 1$, $y' = 2y + 1$ et $r = x^2 - y^2 - N$.

Entrée : N entier composé impair.
Sortie : Deux facteurs non triviaux de n.

$x' := 2\left\lceil \sqrt{N} \right\rceil + 1$, $y' := 1$, $r := \left\lceil \sqrt{N} \right\rceil^2 - N$
Tant que $r \neq 0$ faire
    Si $r < 0$ alors $r := r + x'$, $x' := x' + 2$
    Sinon $(r > 0)$ $r := r - y'$, $y' := y' + 2$
Fin tant que
Retourner les deux facteurs $(x' - y')/2$ et $(x' + y' - 2)/2$

La méthode ρ de Pollard [103]

Soit E un ensemble de cardinal fini m. Soit $(x_n)_{n \in \mathbb{N}}$ est une suite récurrente d'ordre 1, c'est-à-dire de la forme

$$x_0 = a, \qquad x_n = f(x_{n-1}) \quad \text{pour tout } n \in \mathbb{N}^* \tag{2}$$

où f est une application de E dans lui-même et $a \in E$. Alors la suite (x_n) est ultimement périodique. Plus précisément, si t est le premier indice pour lequel il existe $q < t$ tel que $x_t = x_q$ alors on a, en posant $c = t - q$, $x_{n+c} = x_n$ pour tout $n \geqslant q$.

Le "paradoxe des anniversaires" (section 3.2) montre que, si le germe a est pris "au hasard" ainsi que la fonction f, les valeurs moyennes de q et c sont majorées par $k\sqrt{m}$ pour une petite constante k. L'algorithme

suivant (Floyd) permet de trouver un multiple de c (en fait l'*épacte* de la suite, i.e. le plus petit multiple de c qui soit supérieur ou égal à q) en itérant la fonction f et en n'utilisant que très peu de mémoire.

Données : l'ensemble E, une fonction $f : E \to E$ et $a \in E$.
Sortie : l'épacte de la suite (x_n) définie par (2).

$e := 0$
$x := a$
$y := a$
Répéter
    $e := e + 1$
    $x := f(x)$
    $y := f(f(y))$
    si $x = y$ alors retourner e (fin de l'algorithme)
Fin répéter

Le nombre de tours de boucle effectués dans cet algorithme est majoré par $q+c$, donc sa complexité moyenne est en $O(\sqrt{m})$. Une amélioration due à Brent de cet algorithme (ainsi qu'une amélioration de l'amélioration) est décrite, par exemple, dans [30].

L'application de ceci à la factorisation peut se résumer ainsi. Soit p un diviseur (inconnu) de l'entier N à factoriser. En choisissant au hasard un élément a de $\mathbb{Z}_N$ et une fonction f de $\mathbb{Z}_N$ dans lui-même on définit une suite (x_n) récurrente d'ordre 1. Appliquons les remarques qui précèdent à la suite $(x_n \bmod p)$. Il existe un indice k, de valeur moyenne en $O(\sqrt{p})$, et tel que $x_k \equiv x_{2k}$ modulo p. Pour cet indice, on a alors $p \mid \mathrm{pgcd}(x_k - x_{2k}, N)$. Sauf malchance rare, on a $x_k \not\equiv x_{2k}$ modulo N et donc $\mathrm{pgcd}(x_k - x_{2k}, N)$ est un facteur non-trivial (très probablement p, s'il est premier) de N. Pour des raisons d'efficacité, la fonction f n'est pas choisie au hasard (ce qui fausse l'analyse de l'algorithme) mais de la forme $x \mapsto x^2 + b \bmod N$ (avec $b \in \mathbb{Z}_N$). Expérimentalement, cette fonction se comporte bien, c'est-à-dire comme on pourrait l'attendre d'une fonction aléatoire, à condition d'éviter les valeurs $b = 0$ et $b = -2$ (à cause de l'identité $(t+1/t)^2 - 2 = t^2 + 1/t^2$). Une forme simple de l'algorithme ρ de Pollard est donc la suivante :

Entrée : un entier composé N, des constantes $a, b \in \mathbb{Z}_N$.
Sortie : un diviseur non trivial (ou échec) de N.

$x := a$
$y := a$
Répéter
    $x := x^2 + b \bmod N$
    $y := y^2 + b \bmod N$
    $y := y^2 + b \bmod N$
    $d := \mathrm{pgcd}(x - y, N)$
Jusqu'à ce que $d > 1$
Si $d = N$ alors retourner "échec"
Retourner d

Si on admet que les fonctions $x \mapsto x^2 + b$ se comportent comme des fonctions aléatoires, la complexité moyenne de l'algorithme est en $O(p^{1/2+\varepsilon})$ où p est le plus petit facteur premier de N, donc en $O(N^{1/4+\varepsilon})$ (pour tout $\varepsilon > 0$).

Le nombre de Fermat $F_8 = 2^{2^8} + 1$ a été factorisé par une adaptation à ce cas particulier de cet algorithme.

La méthode $p - 1$ de Pollard [102]

Le petit théorème de Fermat a la conséquence suivante. Soit a un entier tel que $\mathrm{pgcd}(a, N) = 1$. Si p est un facteur premier de N et E est un multiple de $p-1$ alors $a^E \equiv 1$ modulo p. Dans ce cas, $\mathrm{pgcd}(a^E - 1, N)$ a de bonnes chances d'être un facteur non trivial de N (à moins bien sûr que E ne soit un multiple de l'indicateur de Carmichael $\lambda(N)$).

Pour optimiser les chances de succès, on choisit un exposant E qui a beaucoup de diviseurs. On peut prendre par exemple le ppcm, ici noté E_B de tous les entiers compris entre 1 et une certaine borne B. Ce ppcm est en fait le produit de tous les nombres de la forme p^k où p est premier et k le plus grand exposant tel que $p^k \leqslant B$. Les diviseurs de E_B sont les entiers fortement B-friables.

Dans la pratique, on décompose $E = p_1 p_2 \cdots p_r$ en produit de facteurs premiers et on calcule $a^E \bmod N$ de proche en proche, en calculant d'abord $a^{p_1} \bmod N$ puis $a^{p_1 p_2} \bmod N \ldots$ On arrête dès qu'on trouve un $\operatorname{pgcd}(a^{p_1 \cdots p_s} - 1, N)$ strictement plus grand que 1.

Entrée : Un entier composé N
Sortie : Un facteur non trivial de N, ou "échec"

$a :=$ valeur quelconque dans l'intervalle $[1, N-1]$
Si $d := \operatorname{pgcd}(a, N) > 1$ alors retourner d
$i := 1$
Tant que $d := \operatorname{pgcd}(a-1, N) = 1$ et $i \leqslant r$ faire
    $a := a^{p_i} \bmod N$
    i := i + 1
Fin tant que
Si $d = 1$ ou $d = N$ alors retourner "échec"
Retourner d

1.1. — **Lemme.** *Soit G un groupe et $a \in G$ un élément d'ordre k. Alors, pour $e \in \mathbb{Z}$, l'ordre de g^e est $k/\operatorname{pgcd}(k, e)$.*

Démonstration — Exercice. □

On peut aussi interpréter l'algorithme de la manière suivante. On choisit un entier a aléatoire dans l'intervalle $[1, N-1]$ et on vérifie que $\operatorname{pgcd}(a, N) = 1$. Si tel est le cas (le cas contraire serait une aubaine) on obtient un élément aléatoire du groupe $\mathbb{Z}_N^*$ et, pour chaque diviseur premier p de N, $a \bmod p$ est un élément du groupe $\mathbb{Z}_p^*$. Lorsqu'on exécute l'instruction $a := a^{p_i} \bmod N$, l'ordre de $a \bmod p$ est divisé par p_i s'il est un multiple de p_i et est inchangé sinon, en vertu de 1.1. Au cours de l'algorithme, l'ordre de $a \bmod p$ est donc décroissant, pour chaque p. Si à un moment donné cet ordre atteint la valeur 1 pour un diviseur p de N alors ce p divise $\operatorname{pgcd}(a-1, N)$. Cela permet de découvrir le facteur p, sauf dans le cas très improbable où l'ordre de $a \bmod N$ atteint 1 au même moment.

L'efficacité de la méthode $p-1$ ne dépend pas directement de la taille des facteurs premiers p de N mais de la nature éventuellement friable des $p-1$ (toutefois, plus p est grand, moins grande est la probabilité que $p-1$ soit B-friable). Si p est un facteur premier de N tel que $p-1$ soit fortement B-friable, la méthode découvre p en environ $B/\ln B$ tours de boucle. Le pire cas pour la méthode $p-1$ est quand $(p-1)/2$ est un nombre premier, pour chaque diviseur p de N. Il faut alors poursuivre l'algorithme jusqu'à $B = \min\{(p-1)/2\}$ ce qui correspond à un coût en $O(N^{1/2})$. Mais dans la pratique, les succès de la méthode $p-1$ sont fréquents et parfois spectaculaires. La méthode $p-1$ admet de nombreux perfectionnements. Pour des compléments, on pourra se reporter à [56] ou [112].

Méthodes dérivées de la méthode $p-1$

La méthode $p-1$ est la première d'une grande famille dont les autres membres sont obtenus en utilisant d'autres groupes à la place de $\mathbb{Z}_p^*$. Ces autres membres fournissent une alternative lorsque les $p-1$ (avec p diviseurs premiers de N) ne sont pas friables.

La méthode $p+1$ de Williams [132] utilise des suites de Lucas. Comme son nom l'indique, elle est efficace pour découvrir les facteurs premiers p tels que $p+1$ soit friable. Il existe aussi des méthodes $p^2 \pm p + 1$, $p^2 + 1$, voire même $\Phi_k(p)$ [14], où Φ_k est le k-ième polynôme cyclotomique.

La méthode ECM [66] de Lenstra utilise le groupe de courbes elliptiques. Contrairement aux méthodes $p-1$ et $p+1$, la méthode ECM n'est pas tributaire de la friabilité de tel ou tel entier, puisque on peut construire des courbes elliptiques sur $\mathbb{F}_p$ de n'importe quel ordre dans l'intervalle $[p+1-2\sqrt{p}, p+1+2\sqrt{p}]$. Son coût pour découvrir un facteur p est en $L_{1/2}(p)$ (voir ci-dessous). Elle a permi en juin 98 à Nancy de découvrir un facteur de 49 chiffres dans un nombre de 132 chiffres, lui-même diviseur de $2^{1071}+1$.

La méthode de Schnorr & Lenstra [118] utilise des groupes de classes de formes quadratiques.

2. Les méthodes à combinaison de congruences

Les méthodes vues jusqu'à présent (sauf ECM) ont un coût en temps exponentiel (de la forme $O(N^c) = O((\exp \ln N)^c)$). L'idéal serait un coût polynomial (en $O((\ln N)^c) = O((\exp \ln \ln N)^c)$) mais semble inaccessible. A partir de la moyenne géométrique des fonctions $\ln N$ et $\ln \ln N$ on construit la fonction L :

$$L(N) = L_{1/2}(N) = \exp\big(\sqrt{(\ln N)(\ln \ln N)}\big),$$

et plus généralement, pour $0 \leqslant t \leqslant 1$, la fonction L_t :

$$L_t(N) = \exp\big((\ln N)^t (\ln \ln N)^{1-t}\big)$$

qui permet d'exprimer les coûts des algorithmes suivants.

La fonction de De Bruijn

On note $\psi(x, y)$ le nombre d'entiers de l'intervalle $[1, x]$ qui sont y-friables. La fonction ψ est connue sous le nom de fonction de De Bruijn. Un résultat important de Pomerance [106] est le suivant.

2.1. — Théorème. *Pour $u > 1$, on a*

$$\psi(x, y) = x u^{-u+o(u)} \qquad \text{en posant } u = \frac{\ln x}{\ln y},$$

dans chaque domaine de la forme $x \geqslant 10$ et $y \geqslant (\ln x)^{1+\varepsilon}$, avec $\varepsilon > 0$.

Appliqué au cas particulier $y = L^a(x)$, on obtient

2.2. — Corollaire. *On a*

$$\frac{\psi\big(x, L(x)^a\big)}{x} = L(x)^{-1/2a+o(1)}.$$

Démonstration — On a d'abord

$$u = \frac{\ln x}{a \ln L(x)} = \frac{1}{a}\sqrt{\frac{\ln x}{\ln \ln x}},$$

donc

$$\begin{aligned} u \ln u &= \frac{1}{a}\sqrt{\frac{\ln x}{\ln \ln x}}\Big(\frac{1}{2}\ln \ln x - \frac{1}{2}\ln \ln \ln x - \ln a\Big) \\ &= \frac{1}{2a}\sqrt{\ln x \ln \ln x}\left(1 - \frac{\ln \ln \ln x - 2 \ln a}{\ln \ln x}\right) = \frac{1}{2a}\sqrt{\ln x \ln \ln x}\big(1 + o(1)\big). \end{aligned}$$

D'après le théorème,

$$\begin{aligned} \frac{\psi\big(x, L(x)^a\big)}{x} &= u^{-u+o(u)} = \exp(-u \ln u)^{1+o(1)} \\ &= \exp\big(\frac{-1}{2a}\sqrt{\ln x \ln \ln x}(1 + o(1))\big)^{1+o(1)} = \Big(L(x)^{-1/2a}\Big)^{1+o(1)} \end{aligned}$$

d'où le corollaire. □

La méthode de Dixon

Cette méthode n'est pas utilisée en pratique, car les suivantes sont plus rapides. Mais elle illustre parfaitement l'ingrédient commun à toutes les méthodes modernes.

Le principe de la méthode de Dixon est le suivant. On fixe une borne B (de taille modeste devant N) et on détermine l'ensemble $\mathcal{B} = \{p_1, \ldots, p_b\}$ des nombres premiers inférieurs à B, appelé *base de factorisation*. Puis, viennent deux grandes étapes.

Dans la première étape, on choisit un grand nombre d'entiers x_i ($i = 1, 2, \ldots$) dans l'intervalle $[1, N-1]$. Pour chacun d'eux, on calcule $a_i = x_i^2 \bmod N$. Puis, par divisions ou calculs de pgcds, on détermine si a_i se factorise sur $\mathcal{B}$, c'est-à-dire se décompose en produit d'éléments de $\mathcal{B}$. Les a_i qui ne sont pas B-friables ne se factorisent pas sur $\mathcal{B}$ et seront ignorés (ainsi que les x_i correspondants). Pour chaque a_i B-friable, la factorisation réussit, et on obtient une relation de la forme

$$x_i^2 \equiv p_1^{e_{i,1}} \cdots p_b^{e_{i,b}} \pmod{N} \tag{3}$$

et on mémorise la valeur de x_i ainsi que le vecteur $e_i = (e_{i,1}, \ldots, e_{i,b})$ décrivant la factorisation de a_i. On poursuit cette étape jusqu'à avoir réussi un peu plus de b factorisations.

La deuxième étape consiste à déterminer, en utilisant des techniques de réduction linéaire sur $\mathbb{F}_2$, des relations de dépendance linéaire modulo 2 entre les vecteurs e_i, c'est-à-dire une famille I telle que $\sum_{i\in I}(e_{i,1}, \ldots, e_{i,b}) = (E_1, \ldots, E_b) \equiv (0, \ldots, 0)$ modulo 2, où $E_j = \sum_{i\in I} e_{i,j}$ pour $1 \leqslant j \leqslant b$. On obtient alors

$$\prod_{i\in I} x_i^2 \equiv p_1^{E_1} \cdots p_b^{E_b}$$

où les exposants E_j sont tous pairs. En posant $x = \prod_{i\in I} x_i$ et $y = \prod_{j=1}^{b} p_j^{E_j/2}$, on a alors (sauf si par malchance $x \equiv \pm y$ modulo N) une relation du type (1).

Voici un exemple simplifié (extrait de [127]). Soit $N = 15770708441$ l'entier à factoriser et choisissons $\mathcal{B} = \{2, 3, 5, 7, 11, 13\}$. Supposons que la première étape fournisse les trois relations

$$\begin{aligned} 8340934156^2 &\equiv 3 \times 7 \pmod{N} \\ 12044942944^2 &\equiv 2 \times 7 \times 13 \pmod{N} \\ 2773700011^2 &\equiv 2 \times 3 \times 13 \pmod{N}. \end{aligned}$$

Leur produit donne

$$(8340934156 \times 12044942944 \times 2773700011)^2 \equiv (2 \times 3 \times 7 \times 13)^2 \pmod{N},$$

et donc, en réduisant modulo N,

$$9503435785^2 \equiv 546^2 \pmod{N}.$$

Il ne reste plus qu'à calculer $\mathrm{pgcd}(9503435785 - 546, N) = 115759$ qui est un diviseur non trivial de N.

Coût de la méthode de Dixon

On choisit comme base de factorisation $\mathcal{B}$ l'ensemble des nombres premiers inférieurs à une borne B de la forme $L(N)^a$. Le cardinal b de B est alors de même ordre de grandeur $L(N)^a$ (car $\pi(L(x)^a) = L(x)^{a+o(1)}$).

D'après 2.2, la probabilité qu'un entier choisi au hasard dans $[1, N-1]$ soit B-friable est de l'ordre de $L(N)^{-1/2a}$ (en fait $L(N)^{-1/2a+o(1)}$ mais à partir de maintenant, **on fera l'abus de notation consistant à ignorer le terme $o(1)$**). Bien que les a_i ne soient pas choisis au hasard, on peut montrer [106], que leur probabilité d'être friables est encore de l'ordre de $L(N)^{-1/2a}$. Si on veut obtenir environ $L(N)^a$ relations (pour que le système linéaire n'ait plus qu'une solution), il faut donc faire de l'ordre de $L(N)^{a+1/2a}$ tentatives. Enfin, le coût en temps de chaque tentative (extraction des facteurs inférieurs à $B = L(N)^a$) est de l'ordre de $L(N)^a$. Au total, le coût de la première phase est donc de l'ordre de $L(N)^{2a+1/2a}$.

Le coût de la résolution d'un système linéaire par réduction de Gauss est fonction cubique de sa taille ; dans le cas qui nous intéresse on a donc un coût de l'ordre de $L(N)^{3a}$. Le coût total de l'algorithme est donc

d'ordre $L(N)^c$ avec $c = \max(3a, 2a + 1/2a)$. Mais $2a + 1/2a$ atteint un minimum égal à 2 pour $a = 1/2$. Pour cette valeur de a, on a $3a < 2$ donc le minimum de c est aussi 2. Le coût (probabiliste) en temps de l'algorithme de Dixon est donc de l'ordre de $L(N)^2$. Son coût en espace correspond au stockage du système linéaire. Pour cette même valeur de $a = 1/2$, il est de l'ordre de $L(N)$.

La méthode de Morrison & Brillhart (fractions continues, [90])

Elle est identique à celle de Dixon, sauf dans la façon de générer des relations du type (3). Au lieu de choisir des carrés modulo N au hasard, on calcule les réduites successives du développement en fraction continue de $\sqrt{N}$. Lorsque la période de ce développement n'est pas assez longue pour fournir suffisamment de réduites, on utilise aussi le développement de $\sqrt{kN}$ pour de petits entiers k. La n-ième réduite p_n/q_n du développement de $\sqrt{kN}$ vérifie

$$\frac{p_n}{q_n} = \sqrt{kN} + \varepsilon \qquad \text{avec } |\varepsilon| \leqslant \frac{1}{q_n^2}.$$

On a donc

$$p_n^2 - q_n^2 kN = O(\sqrt{N}). \tag{4}$$

Il suffit donc de calculer $a_n = p_n^2 - kNq_n^2$ et de tenter de le factoriser comme dans la méthode de Dixon. Le fait que les a_n ne sont pas tous positifs est aisément résolu en adjoignant l'élément -1 à $\mathcal{B}$. De plus, si $p \mid a_n$ alors kN est un carré modulo p. Donc, on peut retirer de $\mathcal{B}$ les nombres premiers p pour lesquels $(kN/p) = -1$ (pour chaque valeur de k utilisée). L'avantage par rapport à la méthode de Dixon est que, d'après (4), les a_n sont de l'ordre de $\sqrt{N}$, et donc bien plus petits. La probabilité qu'ils soient friables est donc meilleure, ce qui améliore le coût de l'algorithme.

Par une analyse semblable à celle de l'algorithme de Dixon, on trouve un coût heuristique de l'ordre de $L(N)^{\sqrt{2}}$ en temps et $L(N)^{1/\sqrt{2}}$ en espace. Le terme *heuristique* correspond au fait suivant. Contrairement à l'algorithme de Dixon, les a_i ne sont pas choisis au hasard et l'algorithme de Morrison & Brillhart est déterministe. Or, il n'est pas prouvé (bien que très probable) que la suite des a_i n'a pas un comportement qui pourrait fausser l'analyse ci-dessus.

La méthode de Morrison-Brillhart à été implantée dans des versions parfois très élaborées et largement utilisée. Elle a en particulier permis la factorisation du nombre de Fermat F_7.

Crible quadratique (Pomerance)

Cette méthode diffère aussi des deux précédentes dans la façon d'obtenir des relations du type (3). Ici, on utilise le polynôme

$$Q(a) = \left(\lfloor\sqrt{N}\rfloor + a\right)^2 - N.$$

Pour $a > 0$ petit, disons $a = O(N^\epsilon)$, $Q(a)$ est petit, de l'ordre de $N^{1/2+\epsilon}$, donc facile à factoriser. Ici encore, seuls les nombres premiers p tels que $(N/p) = 1$ sont susceptibles de diviser $Q(a)$. On peut donc retirer les autres de la base de factorisation $\mathcal{B}$.

Cette façon de générer des relations du type (3) est plus simple que celle utilisant les fractions continues. En contrepartie, les résidus sont légèrement plus grands, et donc plus difficiles à factoriser. Mais au lieu de les factoriser tous et individuellement, on utilise un crible (*sieve*).

Pour cela, on commence par préparer un grand tableau T indexé sur a pour $1 \leqslant a \leqslant A$, où l'on range une approximation grossière de $\ln(Q(a))$. Pour p premier tel que $(N/p) = 1$ et inférieur à une borne B, on calcule les deux racines a_p et b_p de Q modulo p. Mieux, pour chaque $p^k \leqslant B$, on "remonte" ces deux racines modulo p en les deux racines a_{p^k} et b_{p^k} modulo p^k. Ainsi, pour tout a, $Q(a)$ est divisible par p^k si et seulement si a est congru à a_{p^k} ou b_{p^k} modulo p^k. En un rapide parcours de T, on retranche $\ln(p)$ aux valeurs stockées aux indices congrus à a_{p^k} ou b_{p^k} modulo p^k. Après avoir traité tous les p^k inférieurs à B, les $Q(a)$ factorisés sont ceux tels que $T(a)$ est proche de 0.

Sous des hypothèses raisonnables, le coût (probabiliste) du crible quadratique est de l'ordre de $L(N)^{3/2\sqrt{2}}$ en temps et $L(N)^{1/\sqrt{2}}$ en espace. Il peut être très légèrement amélioré en utilisant des méthodes de réduction linéaire plus rapides que celle de Gauss.

MPQS (Crible quadratique utilisant plusieurs polynômes)

D'autres polynômes peuvent jouer un rôle semblable au précédent. Soit

$$Q(X) = AX^2 + 2BX + C \qquad \text{où } N \text{ divise } B^2 - AC.$$

En effet, pour $a \in \mathbb{Z}$, $AQ(a) \equiv (Aa + B)^2$ modulo N. Pour que $|Q(a)|$ soit petit sur un intervalle de taille donnée $2M$ assez grande, on choisit $B^2 - AC > 0$ et on centre l'intervalle $[-B/A - M, B/A + M]$ autour des deux racines. De plus, $|Q(a)|$ est du même ordre de grandeur aux bords de l'intervalle qu'en son centre quand

$$\frac{(AM)^2 + AC - B^2}{A} \quad \text{est de l'ordre de} \quad -\frac{B^2 - AC}{A}$$

c'est-à-dire quand A est de l'ordre de

$$\frac{\sqrt{2(B^2 - AC)}}{M}.$$

On montre alors que si $B^2 - AC$ vaut exactement N, alors le maximum de $|Q(a)|$ sur $[-B/A - M, B/A + M]$ est proche de $M\sqrt{N/2}$.

Cette version du crible quadratique est restée longtemps la meilleure méthode pratique de factorisation. La méthode NFS (ci-dessous) est asymptotiquement meilleure mais n'était jusqu'à récemment pas utilisable en pratique pour des nombres quelconques. La méthode MPQS a gardé longtemps sa suprématie sur NFS pour des nombres sans forme particulière.

NFS (crible sur corps de nombres)

Cette méthode, apparue à partir de 1988, est due à Pollard [105]. Elle utilise un corps de nombres K (de petit degré) et un sous-anneau $\mathbb{Z}[\theta]$ de l'anneau des entiers de K, ainsi qu'un morphisme d'anneaux $\phi : \mathbb{Z}[X] \to \mathbb{Z}$ tel que $c = \phi(X)$ soit une racine modulo N du polynôme minimal de θ. De la sorte, ϕ induit par passage au quotient un morphisme (encore noté ϕ) de $\mathbb{Z}[\theta]$ dans $\mathbb{Z}/N\mathbb{Z}$.

Cette méthode est dans son principe semblable aux méthodes précédentes. Toutefois, les relations du type (3) sont remplacées par des relations du type

$$\phi(\alpha_i) \equiv a_i \pmod{N}.$$

où les α_i sont des éléments de $\mathbb{Z}[\theta]$. On ne se contente plus de factoriser les a_i suivant une base de factorisation, il faut aussi factoriser les α_i dans l'anneau $\mathbb{Z}[\theta]$ en utilisant une autre base de factorisation formée d'irréductibles (et d'inversibles) de $\mathbb{Z}[\theta]$ (et encore, l'anneau $\mathbb{Z}[\theta]$ n'est pas nécessairement principal, ce qui complique un peu plus les choses). Par des méthodes d'algèbre linéaire sur $\mathbb{F}_2$, on détermine une famille I telle que $\prod_{i\in I} \alpha_i = \beta^2$ et $\prod_{i\in I} a_i = y^2$, avec $\beta \in \mathbb{Z}[\theta]$ et $y \in \mathbb{Z}$. On a alors

$$\phi(\beta)^2 = \phi\Big(\prod_{i\in I} \alpha_i\Big) = \prod_{i\in I} a_i = y^2 \quad \text{dans } \mathbb{Z}/N\mathbb{Z}.$$

En posant $x = \phi(\beta)$, c'est une relation du type (1) (sauf si par malchance $x \equiv \pm y$ modulo N).

Le coût de cet algorithme est de l'ordre de $L_{1/3}(N)^c$ pour une petite constante c (proche de 2). Asymptotiquement, c'est donc la meilleure méthode de factorisation connue.

Dans la pratique, on distingue deux versions de cet algorithme. Le *Special* (SNFS) lorsqu'on l'applique à des nombre d'une forme particulière pour lesquels on peut choisir des paramètres simples facilitant le calcul (en particulier K et ϕ), et le *General* (GNFS) lorsqu'on l'applique à des nombres qui n'ont pas de forme particulière exploitable.

La version SNFS a été la première à se développer. L'algorithme NFS a d'ailleurs fait sa renommée en factorisant le nombre de Fermat F_9. D'autre part, un nombre de Mersenne $2^{751} - 1$ de 227 chiffres décimaux à été factorisé en produit de trois facteurs premiers (de 66, 67 et 94 chiffres) le 22 janvier 2002 au CWI d'Amsterdam.

Désormais, la version GNFS est redoutable aussi. Elle a par exemple permis le 2 février 99 (encore au CWI) de trouver la factorisation du nombre RSA-140 (nombre de 140 chiffres proposé par la compagnie RSA, voir `http://www.rsa.com/rsalabs/html/factoring.html`) :

$$\begin{aligned}
&21290246318258757547497882016271517497806703963277216278233383215381 94\\
&\qquad 99840564959113665738530219183167831073879953172308895692308734419364 71\\
&\qquad\qquad =\\
&33987174230284385545301236276138758356339864959695974234909293027714 79\\
&\times \qquad 62642001874012850961516549482644422193020371786235090191111660653946049.
\end{aligned}$$

Depuis (en janvier 2002), RSA-158 a été aussi factorisé par GNFS.

Chapitre 16

Logarithme discret

Rappelons l'énoncé du problème du logarithme discret, dont dépend le cryptosystème d'El Gamal :

0.1. — Problème. Soient $\mathbb{F}_q$ un (grand) corps fini et g une racine primitive dans $\mathbb{F}_q$. Le *problème du logarithme discret* est celui de trouver, étant donné $A \in \mathbb{F}_q^*$, l'entier a (parfois noté $\log_g A$) vérifiant

$$g^a = A \qquad \text{avec } 0 \leqslant a < q-1.$$

La difficulté de ce problème semble comparable à la celle de la factorisation. Il est à noter que tout progrès réalisé dans la résolution de l'un de ces deux problèmes à été suivi d'un progrès similaire dans la résolution de l'autre.

Le problème du logarithme discret peut bien sûr se généraliser à n'importe quel groupe cyclique (ou sous-groupe cyclique d'un groupe quelconque) :

0.2. — Problème. Soient G un groupe cyclique de générateur g et d'ordre n, ainsi qu'un élément $A \in G$. Le problème du *logarithme discret généralisé* consiste à trouver l'entier a (parfois noté $\log_g A$) tel que $0 \leqslant a < n$ et $g^a = A$.

Méthode naïve

Une méthode naïve pour résoudre ce problème (généralisé ou non) consiste à calculer les puissances 1, g, $g^2, \ldots$ de g par une multiplication par g à chaque tour, en comparant chaque puissance obtenue à A. Le nombre de tours de boucle effectués est le logarithme cherché. Le coût de cette méthode est en $O(n)$ opérations dans G. Elle n'est donc utilisable que dans de petits groupes (mais toutefois utile dans ce cas).

1. Méthodes élémentaires

Pas de bébé, pas de géant (*baby step, giant step*)

Cette méthode, due à Shanks [121], utilise un ordre total sur G. Elle réalise un compromis entre le coût en temps et le coût en mémoire.

• On pose $m = \lceil \sqrt{n} \rceil$, c'est la longueur du pas de géant. Celle du pas de bébé est 1.

• On calcule les puissances $1, g^m, g^{2m}, \ldots, g^{(m-1)m}$ de g^m et on classe au fur et à mesure dans une liste les couples (j, g^{jm}) obtenus, suivant les valeurs croissantes de g^{jm}.

• Pour chaque entier i de 0 à (au plus) $m-1$, on calcule Ag^i (une multiplication par g suffit à chaque étape) et on recherche si Ag^i se trouve dans la liste triée des puissances de g^m. On interrompt cette étape dès qu'on a trouvé i_0, j_0 tels que $Ag^{i_0} = g^{j_0 m}$.

• Le logarithme cherché est $j_0 m - i_0 \bmod (q-1)$.

Le coût en mémoire est dominé par celui du stockage de la liste des puissances de g^m. Il est en $O(\sqrt{n} \ln n)$. Pour construire la liste triée (deuxième point), il faut $O(\sqrt{n})$ opérations dans G et $O(m \ln m) = O(\sqrt{n} \ln n)$ comparaisons. Le coût du troisième point est aussi en $O(\sqrt{n})$ opérations dans G et $O(\sqrt{n} \ln n)$ comparaisons. Si on admet que le coût d'une opération dans G est supérieur à $\ln n$, le terme dominant du coût total est de $O(\sqrt{n})$ opérations dans G.

La méthode de Pohlig-Hellman [101]

1.1. — Proposition. *Soient G un groupe cyclique de générateur g et d'ordre n, ainsi qu'un élément $A \in G$. Soit a l'entier tel que $0 \leqslant a < n$ et $g^a = A$. Pour chaque diviseur r de n, on peut déterminer le reste modulo r de a en $O(r + \ln n)$ opérations dans G et $O(r)$ comparaisons dans G.*

Démonstration — En guise de démonstration, voici le procédé utilisé pour calculer $a \bmod r$. Calculons $h = g^{n/r}$ et $B = A^{n/r}$. Le sous-groupe des racines r-ièmes de l'unité dans G est engendré par h et B y appartient. En calculant les puissances h^i pour $0 \leqslant i < r$, on peut va trouver un entier k tel que $B = h^k$. On a alors

$$h^k = B = A^{n/r} = g^{an/r} = h^a.$$

Puisque h est d'ordre r dans G, on a $a \equiv k$ modulo r. Le coût du calcul de h et B est en $O(\ln n)$ opérations dans G et celui du calcul des h^i est en $O(r)$ opérations dans G. Le coût en comparaisons dans G est clairement r. □

1.2. — Proposition. *Soient G un groupe cyclique de générateur g et d'ordre n, ainsi qu'un élément $A \in G$. Soit a l'entier tel que $0 \leqslant a < n$ et $g^a = A$. Pour chaque diviseur de la forme r^e de n, on peut déterminer le reste modulo r^e de a en $O(r + e \ln n)$ opérations dans G et $O((r + e) \ln r)$ comparaisons dans G.*

Démonstration — Voici le procédé utilisé. Pour $e = 1$, on utilise la méthode de 1.1. Pour $e \geqslant 2$, on suppose que l'on vient de calculer $a \bmod r^{e-1}$. Calculons $A' = Ag^{-(a \bmod r^{e-1})}$ et $B' = A'^{n/r^e}$. On a donc

$$A' = g^{\lfloor a/r^{e-1} \rfloor r^{e-1}} \qquad \text{et} \qquad B' = g^{\lfloor a/r^{e-1} \rfloor n/r} = h^{\lfloor a/r^{e-1} \rfloor}$$

où $h = g^{n/r}$ comme dans 1.1. B' est donc une racine r-ième de l'unité et, comme dans 1.1, on détermine k' tel que $B' = h^{k'}$. On a alors $k' \equiv \lfloor a/r^{e-1} \rfloor$ modulo r ou encore $a \equiv k' r^{e-1} + (a \bmod r^{e-1})$ modulo r^e. Pour évaluer le coût, supposons que les puissances de h sont mémorisées et triées dans une liste. Au coût de 1.1, s'ajoutent donc $O(r \ln r)$ comparaisons pour la constitution de cette liste. Ensuite, à chaque itération sur e, il faut $O(\ln r)$ comparaisons (puisque la liste est triée) pour trouver k'. D'où le total de $O((e + r) \ln r)$ comparaisons. Le coût en opérations dans G est celui de 1.1 auquel s'ajoute celui du calcul des A' et B'. □

La méthode de Pohlig-Hellman est efficace lorsque n est friable. Elle consiste à factoriser $n = p_1^{e_1} \cdots p_s^{e_s}$, à calculer le logarithme modulo chacun des $p_i^{e_i}$ et enfin à utiliser le théorème chinois. Soit p le plus grand des p_i, on a $s = O(\ln n)$ et $\sum_{i=1}^{s} e_i = O(\ln n)$. Si on ne compte pas le coût de la factorisation, on obtient un algorithme de calcul du logarithme discret en $O((p + \ln n) \ln n)$ opérations dans G où p est le plus grand facteur premier de n. En combinant avec les idées de la méthode "pas de bébé, pas de géant", on obtient un coût en $O\big((\sqrt{p} + \ln n) \ln n\big)$.

Lorsqu'on connaît la factorisation ou seulement une factorisation partielle de n, l'idée directrice de la méthode de Pohlig-Hellman peut profiter à tous les algorithmes qui suivent dans ce chapitre. On peut ramener ainsi le calcul d'un logarithme dans un groupe d'ordre n au calcul de plusieurs logarithmes dans des groupes plus petits, d'ordre divisant n.

La méthode ρ

Cette méthode est due à Pollard [104] et est apparentée à la méthode de factorisation du même nom. Son coût probabiliste (en temps) est aussi en $O(\sqrt{n})$. Son coût en mémoire est faible.

On choisit deux entiers $u_0, v_0 \in [0, n-1]$ et une fonction $f : G \to G$ au comportement erratique. On pose alors

$$A_0 = g^{u_0} A^{v_0} \qquad \text{et} \qquad A_i = g^{u_i} A^{v_i} = f(A_{i-1}) \quad \text{pour } i = 1, 2, \ldots \tag{1}$$

Pour construire la fonction f, Pollard propose de partitionner G en trois parties G_1, G_2 et G_3 de cardinal voisin et de poser

$$f(P) = \begin{cases} gP & \text{si } P \in G_1, \\ AP & \text{si } P \in G_2, \\ P^2 & \text{si } P \in G_3. \end{cases}$$

De la même manière que dans la méthode ρ de factorisation, on cherche un coïncidence $A_{2e} = A_e$. Pour peu que $v_{2e} - v_e$ soit inversible modulo n, on a alors

$$a = -\frac{u_{2e} - u_e}{v_{2e} - v_e}.$$

La méthode des kangourous

C'est une variante de la méthode ρ, due aussi à Pollard [104]. Elle est aussi appelée méthode λ. Elle est meilleure que la méthode ρ lorsqu'on sait que le logarithme a cherché se trouve dans un intervalle connu. On peut alors supposer que cet intervalle est de la forme $[0, b[$.

Il faut deux kangourous : un sauvage et un apprivoisé. Le kangourou apprivoisé est lâché au point b et on le laisse bondir quelque temps en l'observant. La longueur d'un bond de kangourou est fonction de l'état du terrain à l'endroit où il prend son appel. Elle est décrite par une fonction $f : G \to [1, c]$ (avec c petit) et de valeur moyenne notée α. Si x_k est la position du kangourou apprivoisé après k bonds, alors $x_0 = g^b$ et $x_{k+1} = x_k g^{f(x_k)}$. On récupère le kangourou quand il a effectué s bonds, avec s suffisamment grand (de l'ordre de $\sqrt{b}$). On note son point d'arrivée (on calcule x_s) ainsi que la distance totale parcourue ($d_1 = \sum_{i=0}^{s-1} f(x_i)$) qui est de l'ordre de $s\alpha$. Puis on creuse un piège recouvert de branchages au point x_s.

Le kangourou sauvage part du point $a \in [0, b[$ et la longueur de ses bonds suit la même règle que pour le kangourou apprivoisé, décrite par la même fonction f. Si y_k est la position du kangourou sauvage après k bonds, on a donc $y_0 = g^a = A$ et $y_{k+1} = y_k g^{f(y_k)}$. On le laisse bondir (on calcule les premiers termes de la suite y_k) en observant la distance totale qu'il parcourt (en faisant la somme des $f(y_k)$). Il suffit que le kangourou sauvage parvienne à un endroit où est passé le kangourou apprivoisé pour que ensuite il suive ses traces, puisque leurs bonds sont décrits par la même fonction f. Si les traces du kangourou apprivoisé sont suffisamment nombreuses, la probabilité que leurs trajectoires fusionnent (d'où le nom de méthode λ) est importante. Si tel est le cas, le kangourou sauvage finit par tomber dans le piège ($y_t = x_s$) après un nombre t raisonnable de bonds. Il a alors parcouru une distance totale $d_2 = \sum_{i=0}^{t-1} f(y_i)$.

Connaissant la distance parcourue par chacun des kangourous ainsi que le point de départ du kangourou apprivoisé, il est facile d'en déduire le point de départ du kangourou sauvage :

$$a = b + d_1 - d_2 \bmod n.$$

Si par malchance (chance pour lui) le kangourou sauvage évite les traces du kangourou apprivoisé, il va dépasser le piège après un nombre de bonds de l'ordre de $b/\alpha + s$. On peut donc stopper le kangourou sauvage dès que le nombre de bonds qu'il a effectués dépasse significativement cette valeur et on le relâche à partir d'un point voisin de a.

Le coût en temps de cette méthode est en $O(\sqrt{b})$. Le coût en mémoire est négligeable.

La recherche parallèle de collisions

C'est encore une variante de la méthode ρ, décrite par van Oorschot et Wiener [98], et adaptée au calcul parallèle. Chaque unité de calcul ($1 \leqslant c \leqslant m$) construit une suite de la forme (1), en utilisant des valeurs initiales $A_0^{(c)}$ différentes mais la même fonction f. Parmi les termes $A_i^{(c)}$ calculés, une petite proportion sont distingués par une propriété facilement testable (par exemple, leur représentation binaire commence par un certain nombre de zéros). Les $A_i^{(c)}$ distingués sont stockés (avec les $u_i^{(c)}$ et $v_i^{(c)}$ correspondants) dans une mémoire commune. Lorsqu'un point distingué est rencontré pour la deuxième fois, on dispose de deux décompositions différentes (sauf malchance) de ce point, obtenues par deux unités de calcul c et d :

$$g^{u_i^{(c)}} A^{v_i^{(c)}} = A_i^{(c)} = A_j^{(d)} = g^{u_j^{(d)}} A^{v_j^{(d)}}.$$

On en déduit alors le logarithme de A.

C'est la méthode la plus rapide actuellement fonctionnant pour n'importe quel groupe, en particulier ceux de courbes elliptiques. Elle est utilisée à grande échelle depuis quelques années pour résoudre les challenges proposée par la société Certicom pour promouvoir la cryptographie publique elliptique. Fin 1999, le dernier challenge résolu est ECC2-97, un logarithme discret sur un groupe, fourni par une courbe elliptique, dont l'ordre est un nombre de 97 bits.

2. Méthodes sous-exponentielles

Ces méthodes ne sont pas utilisables pour un groupe quelconque. Elles fonctionnent dans le cas des corps finis. Elles ont été introduites par Adleman [1] pour le cas d'un corps premier $\mathbb{F}_p$. C'est ce cas relativement simple que nous décrivons ici.

Méthode d'Adleman

La méthode d'Adleman comporte deux phases. La première est la plus coûteuse mais n'est effectuée qu'une fois pour toutes lorsqu'on veut calculer plusieurs logarithmes dans le même corps.

Voici d'abord la première phase.

• On choisit un ensemble $\mathcal{B} = \{p_1, p_2, \ldots, p_b\}$ de petits nombres premiers, appelé *base de factorisation*.

• On choisit au hasard des entiers a_i dans l'intervalle $[1, p-2]$ pour $i = 1, 2, \ldots$, et on calcule les entiers $A_i = g^{a_i} \bmod p$. On tente de factoriser chacun des A_i en produit d'éléments de $\mathcal{B}$. Lorsque la factorisation échoue (parce que A_i n'est pas B-friable), on ignore le couple (a_i, A_i).

• Pour chaque A_i dont la factorisation réussit, on obtient une relation du type

$$g^{a_i} \equiv p_1^{e_{i,1}} p_2^{e_{i,2}} \cdots p_b^{e_{i,b}} \pmod{p}$$

et donc une relation de dépendance linéaire (modulo $p-1$) entre les logarithmes des p_i :

$$a_i \equiv e_{i,1} \log_g(p_1) + e_{i,2} \log_g(p_2) + \cdots + e_{i,b} \log_g(p_b) \pmod{p-1}.$$

On mémorise cette relation, c'est-à-dire la valeur de a_i ainsi que le vecteur $(e_{i,1}, \ldots, e_{i,b})$. On poursuit cette étape jusqu'à avoir accumulé un nombre de relations un peu supérieur à b.

• On résout le système linéaire obtenu sur l'anneau $\mathbb{Z}/(p-1)\mathbb{Z}$. Si on a accumulé suffisamment de relations, on trouve une solution unique pour les logarithmes des p_i. On connaît donc désormais chacun des $\log_g p_i$.

Vient ensuite la deuxième phase, pour le calcul final de $\log_g A$. On choisit au hasard un entier s dans l'intervalle $[0, p-2]$. On tente alors de factoriser $A' = Ag^s \bmod p$ dans la base $\mathcal{B}$. Si A' est B-friable, on obtient des entiers $e_1, \ldots, e_b$ tels que

$$A' \equiv p_1^{e_1} p_2^{e_2} \cdots p_b^{e_b} \pmod{p}$$

et donc

$$a + s \equiv e_1 \log_g(p_1) + e_2 \log_g(p_2) + \cdots + e_b \log_g(p_b) \pmod{p-1}.$$

On détermine alors facilement la valeur de $a = \log_g A$. Si au contraire A' n'est pas B-friable alors on recommence la deuxième phase avec une autre valeur de s.

En adaptant l'analyse du coût de la méthode de factorisation de Dixon, on trouve que le coût en temps de la méthode d'Adleman est en $O(L(p)^2)$ et que son coût en espace est en $O(L(p))$.

Exemple

Voici un exemple simplifié (extrait de [84]). Soit $p = 229$ et $g = 6$ (c'est bien un élément d'ordre 228 modulo p). Choisissons $\mathcal{B} = \{2, 3, 5, 7, 11\}$ comme base de factorisation.

La première phase peut fournir (par exemple) les six relations suivantes :

$$\begin{aligned}
g^{100} \bmod p &= 180 = 2^2 \cdot 3^2 \cdot 5 \\
g^{18} \bmod p &= 176 = 2^4 \cdot 11 \\
g^{12} \bmod p &= 165 = 3 \cdot 5 \cdot 11 \\
g^{62} \bmod p &= 154 = 2 \cdot 7 \cdot 11 \\
g^{143} \bmod p &= 198 = 2 \cdot 3^2 \cdot 11 \\
g^{206} \bmod p &= 210 = 2 \cdot 3 \cdot 5 \cdot 7.
\end{aligned}$$

On obtient donc le système linéaire

$$\begin{cases} 100 \equiv 2\log_g 2 + 2\log_g 3 + \log_g 5 \\ 18 \equiv 4\log_g 2 + \log_g 11 \\ 12 \equiv \log_g 3 + \log_g 5 + \log_g 11 \\ 62 \equiv \log_g 2 + \log_g 7 + \log_g 11 \\ 143 \equiv \log_g 2 + 2\log_g 3 + \log_g 11 \\ 206 \equiv \log_g 2 + \log_g 3 + \log_g 5 + \log_g 7 \end{cases} \pmod{228}$$

dont la résolution donne

$$\log_g 2 = 21, \quad \log_g 3 = 208, \quad \log_g 5 = 98, \quad \log_g 7 = 107, \quad \log_g 11 = 162.$$

Si on veut calculer le logarithme de $A = 13$, on peut par exemple choisir $s = 77$ puisque dans ce cas $Ag^s \bmod 229 = 147 = 3 \cdot 7^2$. Donc

$$\log_g 13 = \log_g 3 + 2\log_g 7 - 77 \bmod 228 = 117.$$

Méthodes dérivées

Adleman et De Marrais [2] ont adapté cette méthode au cas d'un corps fini $\mathbb{F}_q$ de cardinal non premier. Le principe est basé sur l'utilisation de la représentation de $\mathbb{F}_q$ comme quotient $F_p[X]/(f)$ de l'anneau des polynômes sur le sous-corps premier $\mathbb{F}_p$. La base de factorisation est alors un ensemble de polynômes irréductibles sur $\mathbb{F}_p$ et de petit degré. On obtient encore un coût en temps en $O(L(q)^2)$.

Coppersmith [36] a donné une version plus rapide utilisable en caractéristique 2. Son coût en temps est en $O(L_{1/3}(q)^c)$ pour une constante $c < 1,6$.

Enfin, les idées de la méthode NFS de factorisation s'adaptent au cas du logarithme discret sur un corps fini quelconque pour donner un algorithme, de coût asymptotique en $O(L_{1/3}(q)^c)$ pour une "petite" constante c.

Pour donner une idée des performances pratiques des ces algorithmes, l'algorithme de Coppersmith a permit en février 2002 de calculer des logarithmes discrets dans le corps $\mathbb{F}_{2^{607}}$. D'autre part, la méthode NFS a permis en avril 2001 (au CELAR à Rennes) de calculer des logarithmes discrets dans $\mathbb{Z}/p\mathbb{Z}$, avec $p = \lfloor 10^{119} \rfloor + 207819$, un nombre premier de 120 chiffres décimaux.

Bibliographie

[1] L. M. Adleman : *A subexponential algorithm for the discrete logarithm problem with applications to cryptography.* Proc. IEEE 20th annual symposium on foundations of computer science, 1979, 55–60.

[2] L. M. Adleman, J. De Marrais : *A subexponential algorithm for discrete logarithms over all finite fields.* Mathematics of Computation, vol. 61, 1993, 1–15.

[3] L. M. Adleman, M. D. Huang : *Recognizing primes in random polynomial time.* Springer-Verlag, Lecture Notes in Mathematics 1512, 1992.

[4] L.M. Adleman, C. Pomerance, R.S. Rumely : *On distinguishing prime numbers from composite numbers.* Annals of Mathematics 117, 1983, 173–206.

[5] G.B. Agnew, T. Beth, R.C. Mullin, S.A. Vanstone : *Arithmetic operations in $GF(2^n)$.* Journal of Cryptology, vol. 6 (n. 1), 1993, 3–13.

[6] M. Agrawal, N. Kayal, N. Saxena : *PRIMES is in P.* preprint, August 2002.

[7] W. Aiello, R. Venkatesan : *Foiling birthday attacks in length doubling transformations.* Lecture Notes in Computer Science 1070 (Eurocrypt'96), 1996, 307–320.

[8] W. Alexi, B. Chor, O. Goldreich, C.P. Schnorr : *RSA and Rabin functions : certain parts are as hard as the whole.* SIAM Journal of Computing, vol. 17, n. 2, April 88, 194–209.

[9] W.R. Alford, A. Granville, C. Pomerance : *There are infinitely many Carmichael numbers.* Annals of Mathematics 140, 1994, 703–722.

[10] A.V. Aho, J.E. Hopcroft, J.D. Hullman : *The Design and Analysis of Computer algorithms.* Addison-Wesley, 1974.

[11] F. Arnault : *The Rabin-Monier theorem for Lucas pseudoprimes.* Mathematics of Computation, n. 218, vol. 66, April 1997, 869–881.

[12] A.O.L. Atkin, F. Morain : *Elliptic curves and primality proving.* Mathematics of Computation, n. 203, vol. 61, July 1993, 29–68.

[13] L. Babaï : *On Lovász lattice reduction and the nearest lattice point problem.* Combinatorica 6, 1986, 1–13.

[14] E. Bach, J. Shallit : *Factoring with cyclotomic polynomials.* Mathematics of Computation, n. 185, vol. 52, Jan. 1989, 201–219.

[15] R. Baillie, S. Wagstaff Jr : *Lucas pseudoprimes.* Mathematics of Computation, n. 152, vol. 35, Oct. 1980, 1391–1417.

[16] H. Beker, F. Piper : *Cipher Systems, the Protection of Communications.* John Wiley and Sons, 1982.

[17] M. Bellare, P. Rogaway : *Optimal Asymetric Encryption — How to encrypt with RSA.* Lecture Notes in Computer Science 950 (Eurocrypt'94), 1994, 92–111.

[18] M. Bellare, P. Rogaway : *The exact security of digital signatures — How to sign with RSA and Rabin.* Lecture Notes in Computer Science 1070 (Eurocrypt'96), 1996, 399-416.

[19] E. Biham, A. Shamir : *Differential Cryptanalysis of the Data Encryption Standard.* Springer-Verlag, 1993.

[20] L. Blum, M. Blum, M. Shub : *A simple unpredictable pseudo-random number generator.* SIAM Journal of Computing, vol. 13, 1986, 364–383 (also in Crypto'82).

[21] M. Blum, S. Goldwasser : *An efficient probabilistic public-key cryptosystem that hides all partial information.* Lecture Notes in Computer Science 196 (Crypto'84), 1985, 289–302.

[22] V.E. Boas : *Machine models computation complexity and number theory.* in: Math. Centre tracts 154, Math. Centrum, Amsterdam, 1982, 7–42.

[23] D. Boneh : *The decision Diffie-Hellman problem.* Lecture Notes in Computer Science 1423 (ANTS-III), 1998, 48–63.

[24] D. Boneh, R. Venkatesan : *Hardness of computing most significant bits in secret keys of Diffie-Hellman and related schemes.* Lecture Notes in Computer Science 1109 (Crypto'96), 129–142.

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

[26] R.P. Brent, J.M. Pollard : *Factorization of the eighth Fermat number.* Mathematics of Computation, n. 154, vol. 36 (1981), 627–630.

[27] D.M. Bressoud : *Factorization and Primality Testing.* Springer-Verlag, 1989.

[28] J. Brillhart, D.H. Lehmer, J. Selfridge, B. Tuckerman, S. Wagstaff : *Factorisations of* $b^n \pm 1$, $b = 2$*, 3, 5, 6, 7, 10, 11, 12, up to high powers.* Contemporary Mathematics 22, A.M.S., Providence R.I., 1983.

[29] B. Chor, O. Goldreich : *RSA/Rabin least significant bits are* $1/2 + 1/\mathrm{poly}(\log n)$ *secure.* Lecture Notes in Computer Science 196 (Crypto'84), 303–313.

[30] H. Cohen : *A Course in Computational Algebraic Number Theory.* Springer-Verlag, GTM 138, 1993.

[31] H. Cohen : *Cryptographie, factorisation et primalité : l'utilisation des courbes elliptiques.* Journée annuelle de la Société Mathématique Française, 1987.

[32] H. Cohen, H.W. Lenstra, Jr. : *Primality testing and Jacobi sums.* Mathematics of Computation, n. 165, vol. 42, Jan. 1984, 297–330.

[33] H. Cohn : *A Second Course in Number Theory.* John Wiley & Sons, 1962.

[34] D.A. Cox : *Primes of the Form* $x^2 + ny^2$. Wiley-Interscience, John Wiley & Sons, 1989.

[35] S.A. Cook : *The complexity of theorem-proving procedures.* Proc. 3rd Annual ACM Symp. on theory of computing, Association for Computing Machinery, New York, 1971, 151–158.

[36] D. Coppersmith : *Fast evaluation of logarithms in fields of characteristic two.* IEEE Transactions on Information Theory, 30 (1984), 587–594.

[37] D. Coppersmith, A.M. Odlyzko, R. Schroeppel : *Discrete Logarithms in* $GF(p)$. Algorithmica 1 (1986), 1–15.

[38] J. Daemen, V. Rijmen : *AES proposal : Rijndael.* `http://www.esat.kuleuven.ac.be/~rijmen/rijndael/rijndaelV2.pdf`.

[39] N.G. de Bruijn : *On the number of uncancelled elements in the sieve of Eratosthenes.* Indag. Math. 12 (1950), 247–256.

[40] D.E.R. Denning : *Cryptography and Data Security.* Addison-Wesley, 1982.

[41] W. Diffie, M.E. Hellman : *New directions in cryptography.* IEEE Transactions on Information Theory, 22 (1976), 644–654.

[42] J.D. Dixon : *Asymptotically fast factorization of integers.* Mathematics of Computation, n. 153, vol. 36 (1981), 255–260.

[43] T. El Gamal : *A public key cryptosystem and signature scheme based on discrete logarithms.* IEEE Trans. on Information Theory, vol IT-31, No 4, July 85, 469–472.

[44] A. Fiat, A. Shamir : *How to prove yourself: practical solutions to identification and signature problems.* Lecture Notes in Computer Science 263 (Crypto'86), 186–194.

[45] R. Fischlin, C.P. Schnorr : *Stronger security proofs for RSA and Rabin bits.* Lecture Notes in Computer Science 1233 (Eurocrypt'97), 267–279.

[46] O. Goldreich : *Modern Cryptography, Probabilistic Proofs and Pseudorandommness.* Algorithms and Combinatorics 17, Springer, 1999.

[47] S. Goldwasser, J. Kilian : *Almost all primes can be quickly certified.* Proceedings 18th annual ACM Symposium on Theory of Computing, 1986, 316–329.

[48] S. Goldwasser, S. Micali : *Probabilistic encryption.* Journal of Computer and System Sciences, 28 (1984), 270–299.

[49] D.M. Gordon : *Discrete logarithms in $GF(p)$ using the number field sieve.* SIAM J. Disc. Math. 6 (1993), no. 1, 124–138.

[50] G.H. Hardy, E.M. Wright : *An Introduction to the Theory of Numbers.* Clarendon Press, 1960–1979 (5th edition).

[51] M.E. Hellman : *An extension of the Shannon theory approach to cryptography.* IEEE Trans. Inf. Theory, IT-23, 289–294, May 1977.

[52] D. Husemöller : *Elliptic Curves.* Springer-Verlag, GTM 111, 1986.

[53] K. Ireland, M. Rosen : *A Classical Introduction to Modern Number Theory.* Springer-Verlag, New York, 2nd edition 1990.

[54] D. Kahn : *The Codebreakers: the story of secret writing.* Macmillan Publ. Co. Inc., New-York, 1977.

[55] A. Karatsuba, Y. Ofman : *Multiplication of multidigits numbers on automata.* Soviet Physics — Doklady, 7 (1963), 595–596.

[56] D.E. Knuth : *The Art of Computer Programming. Tome 2 : Semi-numerical algorithms.* Addison-Wesley, 1973.

[57] D.E. Knuth, L.T. Pardo : *Analysis of a simple factorization algorithm.* Theoretical computer science, Vol. 3 (1976), 321–348.

[58] N. Koblitz : *A Course in Number Theory and Cryptography.* Springer-Verlag, 1987.

[59] N. Koblitz : *Introduction to Elliptic Curves and Modular Forms.* Springer-Verlag, GTM 97, 1984.

[60] C.K. Koç, T. Acar : *Montgomery multiplication in $GF(2^k)$.* Design, Codes and Cryptography, 14 (1), 1998, 57–69.

[61] A.N. Kolmogorov : *Three approaches to the quantitative definition of information.* Problemy Peredachi Informatsii, vol 1 (1965), n. 1, 3–11.

[62] A.G. Konheim : *Cryptography, A Primer.* John Wiley and Sons, 1981.

[63] E. Kranakis : *Primality and Cryptography.* John Wiley & sons/Teubner, 1986.

[64] J.C. Lagarias : *Pseudo-random number generators in cryptography and number theory.* In *Cryptology and Computational Number Theory*, AMS Society, 1990, 115–143.

[65] A.K. Lenstra, H.W. Lenstra : *The Development of the Number Field Sieve.* Lecture Notes in Mathematics, vol. 1554, Springer-Verlag, 1993.

[66] H.W. Lenstra, Jr. : *Factoring integers with elliptic curves.* Annals of Mathematics 126, 1987, 649–673.

[67] A.K. Lenstra, H.W. Lenstra, jr, M.S. Manasse, J.M. Pollard : *The factorization of the ninth Fermat number.* Mathematics of Computation, n. 203, vol 61, July 1993, 319–349.

[68] A.K. Lenstra, E.R. Verheul : *Selecting cryptographic key sizes.* Journal of Cryptology, vol. 14 (n. 4), 2001, 255–293.

[69] A.K. Lenstra, E.R. Verheul : *The XTR public key system.* Lecture Notes in Computer Science 1880 (Crypto'00), 2000, 1–20.

[70] W.J. LeVeque : *Topics in Number Theory (Vols. 1 et 2).* Addison-Wesley Publishing Company, Inc. Reading, Third printing 1965.

[71] R. Lidl, H. Niederreiter : *Finite fields.* Cambridge University Press, Cambridge 1984.

[72] D.L. Long, A. Wigderson : *The discrete logarithm hides $O(\log n)$ bits.* SIAM Journal of Computing, vol. 17, n. 2, April 88, 363–372.

[73] M. Luby, C. Rackoff : *How to construct pseudorandom permutations from pseudorandom fonctions.* SIAM Journal of Computing, vol. 17, n. 2, April 88, 373–386.

[74] R.J. McEliece : *A public-key cryptosystem based on algebraic coding theory.* DSN Progress Report, Jet Propulsion Laboratory, Pasadena, 1978, 42–44.

[75] Z. Manna : *Mathematical Theory of Computation.* McGraw-Hill, 1974.

[76] P. Martin-Löf : *The definition of random sequences.* Information and Control, vol. 9 (1966), 602–619.

[77] M. Matsui : *Linear cryptanalysis method for DES cipher.* Lectures Notes in Computer Science 765 (Eurocrypt'93), Springer Verlag, 386–397.

[78] U.M. Maurer : *A universal statistical test for random bit generators.* Journal of Cryptology, vol 5 (1992), 89–105.

[79] U.M. Maurer : *A simplified and generalized treatment of Luby-Rackoff pseudorandom permutation generators.* Lecture Notes in Computer Science 658 (Eurocrypt'92), Springer Verlag, 239–255.

[80] U.M. Maurer : *Towards the equivalence of breaking the Diffie-Hellman protocol and computing discrete logarithms.* Lecture Notes in Computer Science 839 (Crypto'94), Springer Verlag, 271–281.

[81] U.M. Maurer : *Fast generation of prime numbers and secure public-key cryptographic parameters.* Journal of Cryptology, vol. 8 (1995), 123–155.

[82] A.J. Menezes : *Elliptic Curve Public Key Cryptosystems.* Kluwer Academic Publishers, 1993.

[83] A. Menezes, I. Blake, X. Gao, R. Mullin, S. Vanstone, T. Yaghoobian : *Applications of Finite Fields.* Kluwer Academic Publishers, 1993.

[84] A.J. Menezes, P.C. Van Oorschot, S.A. Vanstone : *Handbook of Applied Cryptography.* CRC Press, 1996.

[85] S. Micali, C.P. Schnorr : *Efficient, perfect polynomial random number generators.* Journal of Cryptology, vol. 3 (n. 3), 1991, 157–172.

[86] G.L. Miller : *Riemann's Hypothesis and tests for primality.* Journal of Computer and System Sciences, 13 (1976), 300–317.

[87] L. Monier : *Evaluation and comparison of two efficient primality testing algorithms.* Theoretical Computer Science, vol. 11, 1980, 97–108.

[88] P. Montgomery : *Modular multiplication without trial division.* Mathematics of Computation, vol. 44 (1985), 519–521.

[89] M.A. Morrison : *A note on primality testing using Lucas sequences.* Mathematics of Computation, vol. 29 (1975), 181–182.

[90] M.A. Morrison, J. Brillhart : *A method of factoring and the factorization of F_7.* Mathematics of Computation, vol. 29 (1975), 183–205.

[91] National Institute of Standards and Technology : *Data Encryption Standard (DES).* FIPS 46-3, 1977, 1999.

[92] National Institute of Standards and Technology : *Secure Hash Standard (SHA-1).* FIPS 180-1, 1995.

[93] National Institute of Standards and Technology : *Digital Signature Standard (DSS).* FIPS 186-2, 2000.

[94] National Institute of Standards and Technology : *Advanced Encryption Standard (AES).* FIPS 197, 2001.

[95] P. Naudin, C. Quitté : *Algorithmique Algébrique.* Masson, 1992.

[96] H. Niederreiter : *Knapsack type cryptosystems and algebraic coding theory.* Problems of Control and Information Theory 15 (2), 1986, 157–166.

[97] B.C. Neuman, T. Ts'o : *Kerberos: An authentication service for computer networks.* IEEE Comm. Magazine, 32 (9), 1994, 33–38.

[98] P.C. van Oorschot, M.J. Wiener : *Parallel collision search with cryptanalytic applications.* Journal of Cryptology, vol 12 (n. 1), 1999, 1–28.

[99] R. Peralta : *Simultaneous security of the discrete log.* Lecture Notes in Computer Science 219 (Eurocrypt'85), LNCS 219, 62–72.

[100] H.C. Pocklington : *The determination of the prime or composite character of large numbers by Fermat's theorem.* Proc. Cambridge Philos. Soc. 18 (1914/6), 29–30.

[101] S.C. Pohlig, M.E. Hellman : *An improved algorithm for computing logarithms over GF(p) and its cryptographic significance.* IEEE Transactions on Information Theory, 24 (1978), 106–110.

[102] J.M. Pollard : *Theorems on factorization and primality testing.* Proc. Cambridge Philos. Soc. 76 (1974), 521–528.

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

[104] J.M. Pollard : *Monte Carlo methods for index computation (mod p).* Mathematics of Computation, vol. 32 (1978), 918–924.

[105] J.M. Pollard : *Factoring with cubic integers.* In [65], 1988–1993, 4–10.

[106] C. Pomerance : *Analysis and comparison of some integer factoring algorithms.* in Computational Methods in Number Theory, part I, 89–139, Mathematisch Centrum, 1982.

[107] C. Pomerance : *Factoring.* in Cryptology and Computational Number Theory, 27–46, American Mathematical Society, 1990.

[108] C. Pomerance, J.L. Selfridge, S.S. Wagstaff : *The pseudoprimes to $25 \cdot 10^9$.* Mathematics of Computation, n. 151, vol. 35, 1980, 1003–1026.

[109] V.R. Pratt : *Every prime has a succinct certificate.* SIAM Journal of Computing, vol. 4, 1975, 214–220.

[110] M.O. Rabin : *Probabilistic Algorithms for testing primality.* Journal of Number Theory, 12 (1980), 128–138.

[111] P. Ribenboim : *The Book of Prime Number Records.* Springer-Verlag, 1988.

[112] H. Riesel : *Prime Numbers and Computers Methods for Factorization.* Birkaüser Boston Inc., 1985.

[113] R.L. Rivest, A. Shamir, L. Adleman : *A method for obtaining digital signature and public key cryptosystems.* Comm. ACM, vol 21, Feb 78, 120–126.

[114] G. Robin : *Algorithmique et Cryptographie.* Mathématiques et Applications 8, S.M.A.I., Ellipses, 1991.

[115] A. Salomaa : *Public-Key Cryptography.* Springer-Verlag, 1990.

[116] P. Samuel : *Théorie Algébrique des Nombres.* Hermann, Collection Méthodes, Paris 1967.

[117] B. Schneier : *Applied Cryptography: Protocols, Algorithms, and Source Code in C.* John Wiley & Sons, New York, 2nd edition, 1996.

[118] C.P. Schnorr, H.W. Lenstra : *A Monte Carlo factoring algorithm with linear storage.* Mathematics of Computation, n. 167, vol. 43 (1984), 289–311.

[119] R.J. Schoof : *Quadratic Fields and factorization.* Computational Methods in Number Theory, part II, H.W. Lenstra, R Tijdeman (ed.). Mathematical Centre Tracts 155, Amsterdam 1982, 235–286.

[120] J.P. Serre : *Cours d'Arithmétique.* Presses Universitaires de France, Collection SUP, 1970.

[121] D. Shanks : *Five number-theoretic algorithms.* Proceedings of the Second Manitoba Conference on Numerical Mathematics, 1972, 51–70.

[122] C.E. Shannon : *Communication theory of secrecy systems.* Bell Syst. Tech. J., vol 28, 1949, 656–715.

[123] J.H. Silverman : *The Arithmetic of Elliptic Curves.* Springer-Verlag, GTM 106, 1986.

[124] J.H. Silverman : *Advanced Topics in the Arithmetic of Elliptic Curves.* Springer-Verlag, GTM 151, 1994.

[125] R.D. Silverman : *The Multiple Polynomial Quadratic Sieve.* Mathematics of Computation, n. 177, vol. 48 (1987), 329–339.

[126] R. Solovay, V. Strassen : *A fast Monte Carlo test for primality.* SIAM Journal on Computing, 6 (1977), 84–85.

[127] D. Stinson : *Cryptography: Theory and Practice.* CRC, 1995.

[128] A.M. Turing : *On computable numbers, with an application to the entscheidungsproblem.* Proc. London Math. Soc., Ser. 2, vol 42, 1937, 230–265. Correction, vol 43, 544-546.

[129] A. Turing : *The Enigma.* Simon and Schuster, New-York, 1983.

[130] U.V. Vazirani, V.V. Vazirani : *Efficient and secure pseudo-random number generation.* Lecture Notes in Computer Science 196 (Crypto'84), 193–202. Also: Proceedings 25th IEEE Symposium on foundations of computer science, 458–463.

[131] M.J. Wiener : *Cryptanalysis of short RSA secret exponents.* IEEE Trans. Inform. Theory 36 (1990), no. 3, 443–551.

[132] H.C. Williams : *A $p+1$ method of factoring.* Mathematics of Computation, vol. 39, July 1982, 225–234.

[133] H.C. Williams, J.S. Judd : *Determination of the primality of N by using factors of $N^2 \pm 1$.* Mathematics of Computation, n. 133, vol. 30, Jan. 1976, 157–172.

[134] A.C. Yao : *Theory and applications of trapdoor functions.* Proceedings of 23rd IEEE symposium on foundations of computer science, 1982, 80–91.