UNIVERSITÉ BORDEAUX 1

IREM D'AQUITAINE

INSTITUT DE RECHERCHE POUR L'ENSEIGNEMENT DES MATHÉMATIQUES

Site Lamartine - 40, rue Lamartine
33400 TALENCE

-=-

Initiation à la cryptologie

Henri COHEN

Michel OLIVIER

Laboratoire A2X
(Arithmétique Algorithmique eXpérimentale)

-=-

Bordeaux, septembre 2000

L'ivresse est dans le nombre

Charles Baudelaire.

Présentation

On ne peut que se réjouir d'une nouvelle introduction de l'Arithmétique dans l'Enseignement secondaire, d'autant qu'elle est accompagnée –faut-il souligner que cela est nouveau dans les programmes et leurs commentaires ?- de vives incitations à explorer avec les élèves quelques uns des domaines les plus significatifs des méthodes et algorithmes arithmétiques (avec des avancées souvent récentes).

La cryptographie, plus généralement la cryptologie, est l'un d'eux.

J'ai demandé à deux éminents spécialistes, les professeurs *Henri Cohen* et *Michel Olivier*, du laboratoire A2X (*Arithmétique Algorithmique eXpérimentale*) de l'Université Bordeaux 1, de bien vouloir assurer un stage de formation continue sur ce sujet. Ils ont accepté : qu'ils trouvent là l'expression renouvelée des remerciements de l'IREM d'Aquitaine.

Ils portent l'entière responsabilité du succès de cette opération et la demande pressante auprès de l'IREM de publication du présent document (a priori document d'accompagnement à l'attention des stagiaires) en est l'une des retombées les plus marquantes.

Si, de la volonté même des auteurs, ce document reste un document d'initiation, il convaincra aisément le lecteur que la cryptologie est résolument irréductible à quelques clichés ou à quelques curiosités...

Pierre-Henri Terracher
Directeur de l'IREM d'Aquitaine.

Université Bordeaux I
U.F.R. Maths-Info

Une introduction à la cryptologie

MICHEL OLIVIER

1. Généralités

Définition 1. *La cryptologie est l'étude des techniques mathématiques liées à la sécurité de l'information, c'est-à-dire la confidentialité, l'intégrité des données, l'authentification d'une entité (personne, ordinateur, carte de crédit, etc ...), et l'authentification de l'origine des données,*
La cryptologie se scinde en deux parties :

La cryptographie est l'étude des moyens de parer aux attaques humaines sur la communication.

La cryptanalyse est la tentative pour un adversaire de déchiffrer un texte chiffré ou pour trouver les secrets du chiffrement.

Ne pas confondre cette théorie avec le codage, ou la théorie des codes, qui traite des problèmes liés à la défaillance physique des canaux de transmissions des données.

L'objectif de la cryptographie est de permettre à deux personnes A (comme Alice) et B (comme Bob) de communiquer à travers un canal de transmission de telle sorte qu'un adversaire O (comme Oscar) ne puisse comprendre le sens du message transmis.

A envoie à B un message chiffré (ciphertext) obtenu à partir du message en clair (plaintext) en utilisant une clef prédéterminée (key), à travers le canal. O ayant capté le message chiffré sur le canal de transmission, ne peut retrouver le message en clair ; seul B, qui reçoit le message chiffré possède un algorithme de déchiffrement lui permettant de reconstituer le message en clair.

Définition 2. *Un système cryptographique (cryptosystem) est un quintuplet* $S = (\mathcal{P}, \mathcal{C}, \mathcal{K}, \mathcal{E}, \mathcal{D})$ *dont les composantes sont définies comme suit :*

Soit $\mathcal{A}$ *un alphabet (un ensemble fini de lettres);* $\mathcal{P}$ *est un ensemble fini de mots construits avec l'alphabet* $\mathcal{A}$ *appelé espace des messages en clair (plaintexts) ;* $\mathcal{C}$ *est un ensemble fini appelé espace des messages chiffrés ou cryptogrammes (ciphertexts) ;* $\mathcal{K}$ *est un ensemble fini appelé espace des clefs (keys) ;* $\mathcal{E}$ *est l'ensemble des*

règles ou algorithmes de chiffrement (encryption rules) ; un élément de $\mathcal{E}$ est une bijection de $\mathcal{P}$ sur $\mathcal{C}$; $\mathcal{D}$ est l'ensemble des règles ou algorithmes de déchiffrement (decryption rules) ; un élément de $\mathcal{D}$ est une bijection de $\mathcal{C}$ sur $\mathcal{P}$; on a la relation suivante entre $\mathcal{E}$ et $\mathcal{D}$: pour tout $K \in \mathcal{K}$ il existe $e_K \in \mathcal{E}$ et $d_K \in \mathcal{D}$ tels que $d_K \circ e_K = \mathrm{id}_{\mathcal{P}}$.

Il y a aujourd'hui deux types de systèmes cryptographiques :
les systèmes dits à clefs privées (private key cryptosystems),
les systèmes dits à clefs publiques (public key cryptosystems).

2. Quelques repères chronologiques et bibliographiques

2.a. Chronologie

de -4000 à 1960 : la cryptographie est militaire et diplomatique
après 1960 : elle entre dans les entreprises et chez les particuliers
1976 : Diffie et Hellman définissent les systèmes à clef publique
1977 : adoption du système DES (à clef privée)
1978 : description du système RSA (à clef publique)
1985 : système El Gamal (à clef publique)
1991 : ISO 9796, standard pour la signature digitale (basé sur RSA)
1994 : DSS, standard pour la signature digitale (basé sur le système El Gamal)
1998 : fin du standard DES

2.b. Bibliographie succinte

D. KAHN, The Codebreakers : the story of secret writing, Macmillan Publishing Company Inc., New-York, 1977.
(Le "roman" de la cryptographie : se lit comme un polar.)
N. KOBLITZ, A course in Number Theory and Cryptography, Springer-Verlag, 1987.
(Les mathématiques dans la cryptographie.)
A. J. MENEZES, P. C. VAN OORSCHOT AND S. A. VANSTONE, Handbook of Applied Cryptography, CRC Press, 1997.
(Pour les cryptographes experts : la science, la loi, les références, etc ...)
NATIONAL BUREAU OF STANDARDS, Data Encryption System (DES), FIPS Publication 46, 1977.
NATIONAL BUREAU OF STANDARDS, Digital Signature Standard (DSS), FIPS Publication 113, 1985.
(Les standards par le bureau des standards américains ... et par conséquent mondiaux.)

R. L. RIVEST, A. SHAMIR AND L. ADLEMAN, A method for obtaining digital signature and public key cryptosystems, Communications ACM, vol. 21, 1978, pages 120–126.
(L'article original sur RSA, le système aujourd'hui universel.)
D. R. STINSON, Cryptography : theory and practice, CRC Press, 1995.
(Un manuel simple pour amateur éclairé ; les pages suivantes en sont fortement imprégnées.)
A. TURING, The enigma, Simon and Schuster, New-York, 1983.
(La célèbre histoire du chiffre allemand durant la seconde guerre mondiale (cf. aussi le film romancé).)

3. L'antiquité de la cryptologie (de -4000 à 1977)

Il s'agit ici de décrire brièvement les principes des systèmes cryptographiques à clef privée utilisés avant 1977.

Protocole. *Le protocole de transmission est le suivant :*

(1) *A et B conviennent secrètement d'une clef $K \in \mathcal{K}$, à l'insu de O.*

(2) *Le message en clair P que A veut transmettre à B est découpé en m mots de n caractères, soit $P = (x_1, \ldots, x_m)$ écrits sur l'alphabet $\mathcal{A}$ tels que pour tout i, $x_i \in \mathcal{P}$.*

(3) *A calcule pour tout i, $y_i = e_K(x_i)$ et envoie à B le message chiffré $C = (y_1, \ldots, y_m)$ à travers le canal.*

(4) *B calcule pour tout i, $x_i = d_K(y_i)$ et reconstitue ainsi le message en clair $P = (x_1, \ldots, x_m)$.*

Notation. On identifie dans la suite l'alphabet usuel $\mathcal{A} = (a, b, c, \ldots, z)$ avec $\mathbb{Z}/26\mathbb{Z}$ par $a = 0$, $b = 1$, $c = 2$, etc ..., $z = 25$.

On donnera à chaque fois les valeurs du quintuplet $\mathcal{S} = (\mathcal{P}, \mathcal{C}, \mathcal{K}, \mathcal{E}, \mathcal{D})$ qui définit le système cryptographique.

3.1. Le chiffrement par décalage (shift cipher)

Utilisé – dit-on – par Jules Cesar durant la guerre des Gaules, avec $K = 3$.
$\mathcal{P} = \mathcal{C} = \mathcal{K} = \mathbb{Z}/26\mathbb{Z}$; pour $K \in \mathbb{Z}/26\mathbb{Z}$ et $x \in \mathbb{Z}/26\mathbb{Z}$, on a $e_K(x) = x + K \mod 26$ et par conséquent $d_K(x) = x - K \mod 26$.

Exercice 1 : déchiffrer les messages $FUBSWRJUDSKLH$ et 12.25.24.20.25.5.2.

Le nombre de clefs possibles étant petit (26), c'est très facile...

3.2. Le chiffrement par substitution ou par flot (substitution cipher)

Le plus utilisé jusqu'à la seconde guerre mondiale.
$\mathcal{P} = \mathcal{C} = \mathbb{Z}/26\mathbb{Z}$; $\mathcal{K} = S_{26}$, le groupe des permutations sur 26 lettres (les éléments de $\mathbb{Z}/26\mathbb{Z}$) ; pour $K \in S_{26}$ et $x \in \mathbb{Z}/26\mathbb{Z}$, $e_K(x) = K(x)$ et $d_K(x) = K^{-1}(x)$, où K^{-1} désigne la permutation inverse de K.

Exercice 2 : Trouver la permutation (au moins une partie) qui a chiffré le mot en clair de l'exercice 1 sous la forme $MDQFERUDPFJNI$.

Il y a $26! \simeq 4.10^{26}$ clefs ; mais cela n'empêche pas la cryptanalyse d'être efficace.

3.3. Le chiffrement affine (affine cipher)

$\mathcal{P} = \mathcal{C} = \mathbb{Z}/26\mathbb{Z}$; $\mathcal{K} = \{(a,b) \in \mathbb{Z}/26\mathbb{Z} \times \mathbb{Z}/26\mathbb{Z} \mid (a,26) = 1\}$; pour $K = (a,b) \in \mathcal{K}$ et $x \in \mathbb{Z}/26\mathbb{Z}$, $e_K(x) = ax + b \mod 26$ et $d_K(x) = a^{-1}(x - b) \mod 26$, où a^{-1} désigne l'inverse de a dans $\mathbb{Z}/26\mathbb{Z}$ (pourquoi cet inverse existe-t-il ?).

Il y a $12.26 = 312$ clefs possibles : c'est trop peu.

Exercice 3 : chiffrer la phrase "cet exercice est stupide" avec la clef $K = (7, 12)$.

3.4. Le chiffrement de Vigenère (Vigenere cipher)

Les chiffrements précédents sont monoalphabétiques, c'est-à-dire que l'on chiffre une lettre x à la fois et que $e_K(x)$ n'est constitué aussi que d'une seule lettre. Le chiffrement de Vigenère est polyalphabétique.
$\mathcal{P} = \mathcal{C} = \mathcal{K} = (\mathbb{Z}/26\mathbb{Z})^m$, où m est un entier naturel positif ; pour $K = (k_1, \ldots, k_m)$ et $x = (x_1, \ldots, x_m) \in \mathcal{P}$, $e_K(x) = (x_1 + k_1, \ldots, x_m + k_m) \mod 26$ et $d_K(x) = (x_1 - k_1, \ldots, x_m - k_m) \mod 26$.

Exercice 4 : Combien y-a-t-il de clefs posssibles ? Pour $m = 3$ et $K = (2, 11, 4)$, déchiffrer le mot "ECCRESICERSMG".

Remarquer que dans un système polyalphabétique le chiffrement d'une lettre isolée prend m valeurs possibles.

3.5. Le chiffrement de Hill (Hill cipher)

C'est un chiffrement polyalphabétique.
$\mathcal{P} = \mathcal{C} = (\mathbb{Z}/26\mathbb{Z})^m$; $K = Gl_m(\mathbb{Z}/26\mathbb{Z})$ groupe des matrices inversibles à m lignes et colonnes à coefficients dans $\mathbb{Z}/26\mathbb{Z}$ (le déterminant de la matrice doit être inversible dans $\mathbb{Z}/26\mathbb{Z}$) ; pour $K \in \mathcal{K}$ et $x = (x_1, \ldots, x_m) \in \mathcal{P}$, $d_K(x) = (x_1, \ldots, x_m)K$ (produit de deux matrices) et $d_K(x) = (x_1, \ldots, x_m)K^{-1}$.

Exercice 5 : Calculer le nombre de clefs possibles (plus dur ; essayer avec $m = 2$). Pour $m = 2$ et $K = \begin{pmatrix} 4 & 3 \\ 9 & 7 \end{pmatrix}$ chiffrer le mot "CRYPTOGRAPHE".

3.6. Chiffrement par permutation

$\mathcal{P} = \mathcal{C} = (\mathbb{Z}/26\mathbb{Z})^m$; $\mathcal{K} = S_m$ groupe des permutations sur m lettres ; pour $\pi \in S_m$ et $x = (x_1, \ldots, x_m) \in \mathcal{P}$,

$$e_\pi(x) = (x_{\pi(1)}, \ldots, x_{\pi(m)}) \text{ et } d_\pi(x) = (x_{\pi^{-1}(1)}, \ldots, x_{\pi^{-1}(m)}).$$

Exercice 6 : si $m = 6$ et $\pi = \begin{pmatrix} 1 & 2 & 3 & 4 & 5 & 6 \\ 4 & 2 & 1 & 5 & 6 & 3 \end{pmatrix}$, déchiffrer le mot "PRCTOYPRGHEA".

C'est un cas particulier du chiffrement de Hill via la matrice de permutation donnée par l'application

$$\begin{aligned} S_m &\to Gl_m(\mathbb{Z}/26\mathbb{Z}) \\ \pi &\mapsto K_\pi = (\delta_{i,\pi(j)})_{1 \leq i,j \leq m}. \end{aligned}$$

3.7. One Time Pad ou système cryptographique de Vernam

Utilisé vers 1917 pour le télégraphe, avec l'alphabet $\mathcal{A} = \{0, 1\}$.

Identique au chiffrement de Vigenère en n'utilisant la clef qu'une seule fois ; on choisit une clef aussi longue que le message à chiffrer ! Le décalage change donc à chaque lettre chiffrée.

3.8. Chiffrement en chaîne (stream cipher)

Dans les chiffrements précédents, les éléments de $\mathcal{P}$ sont chiffrés avec la même clef. L'idée ici est d'avoir une clef variable, dépendant de la place de $x \in \mathcal{P}$ dans le message en clair.

Si $x = x_1 x_2 x_3 \ldots$ est le message en clair ($x_i \in \mathcal{P}$) et si $K \in \mathcal{K}$, on définit une suite de clefs correspondantes $z_1 z_2 z_3 \ldots$ et la suite des règles de chiffrement $e_{z_1} e_{z_2} e_{z_3} \ldots$ au moyen d'une fonction $z_i = f_i(K, x_1, x_2, \ldots, x_{i-1})$. On chiffre alors par la formule $y_i = e_{z_i}(x_i)$ pour obtenir le message chiffré $y_1 y_2 y_3 \ldots$.

En résumé, pour chiffrer le message, on calcule dans l'ordre $z_1, y_1, z_2, y_2, \ldots$. Pour déchiffrer, on calcule dans l'ordre $z_1, x_1, z_2, x_2, \ldots$.

Exemple : On se donne $K = (k_1, \ldots, k_m) \in (\mathbb{Z}/26\mathbb{Z})^m$ et $(c_0, \ldots, c_{m-1}) \in (\mathbb{Z}/26\mathbb{Z})^m$ fixés et pour tout i entre 1 et m on pose $z_i = k_i$ et $z_{i+m} = \sum_{j=0}^{m-1} c_j z_{i+j}$ mod 2. La clef est donc ici $(k_1, \ldots, k_m, c_0, \ldots, c_{m-1})$.

Exercice 7 : La clef est "A la recherche du temps perdu" de Marcel Proust, édition de La Pleïade, et les nombres (date du jour de réception,2,10). Chiffrer le message

"C'est une méthode fatigante mais sure pour les espions qui sont dans le froid". Pour chiffrer, on ouvre le livre à la page dont le numéro est le numéro du jour de réception du message (par exemple le 151-ème jour de l'année) et on commence au deuxième paragraphe : pour la lettre de rang i du message chiffré (dans l'ordre), on calcule la clef de chiffrement en considérant la i-ème lettre du paragraphe codée modulo 26 en décalant de $+10$; soit a_i le nombre obtenu. On chiffre alors la lettre du message en calculant $x_i + a_i \mod 26$.

Vous devriez aller voir des vieux films d'espionnage : vous sauriez cela !

Remarque : Un chiffrement en chaîne est donc un 7-uplet

$$\mathcal{S} = (\mathcal{P}, \mathcal{C}, \mathcal{K}, \mathcal{L}, \mathcal{F}, \mathcal{E}, \mathcal{D})$$

où :

$\mathcal{P}$ est l'ensemble des messages en clair,

$\mathcal{C}$ est l'ensemble des messages chiffreés,

$\mathcal{K}$ est l'ensemble des clefs,

$\mathcal{L}$ est un ensemble fini : l'alphabet de la chaîne des clefs,

$\mathcal{F} = (f_1, f_2, f_3, \ldots)$ est le générateur de la chaîne des clefs,

$\mathcal{E}$ est l'ensemble des règles de chiffrement,

$\mathcal{D}$ est l'ensemble des règles de déchiffrement.

Pour tout $i \geq 1$, f_i est une application $f_i : \mathcal{K} \times \mathcal{P}^{i-1} \to \mathcal{L}$, et, pour tout $z \in \mathcal{L}$ il existe une règle de chiffrement $e_z \in \mathcal{E}$ et une règle de déchiffrement $d_z \in \mathcal{D}$ telles que $e_z : \mathcal{P} \to \mathcal{C}$ et $d_z : \mathcal{C} \to \mathcal{P}$ avec $d_z \circ e_z = \mathrm{id}_{\mathcal{P}}$.

Un chiffrement en chaîne est dit synchrone si f_i ne dépend pas de $\mathcal{P}$, et est dit périodique de période d si pour tout $i \geq 1$ on a $z_{i+d} = z_i$.

Par exemple, le chiffrement de Vigenère est périodique de période $d = m$.

3.9. La machine Enigma

La meilleure façon de casser un chiffrement – c'est à dire la meilleure cryptanalyse – c'est de trouver la machine automatique qui chiffre. C'est ce qui est arrivé aux alliés durant la seconde guerre mondiale (à bord d'un sous-marin allemand). Le secret a consisté alors à garder secret le fait qu'on avait trouvé la machine à chiffrer, permettant ainsi d'envoyer des messages chiffrés pour intoxiquer l'ennemi d'alors.

Cette machine était une machine à rotors (disques enfilés sur un axe et tournant séparément). Chaque rotor était muni de 26 contacts sur chaque face (un pour chaque lettre) et était cablé de façon interne réalisant ainsi une permutation de $\mathbb{Z}/26\mathbb{Z}$. La permutation changeait en une rotation conjuguée au sens de S_{26} lorsque le rotor tournait. Il y avait trois rotors, permettant de composer trois permutations.

A chaque lettre, les rotors tournaient de façon incrémentale, 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 $26.25.26 = 16900$. En plus, une permutation supplémentaire était réalisée par un tableau à connecteurs. La clef était donc constituée de la configuration de ce tableau et de la position initiale des rotors .... OUF ...

(Si vous n'avez rien compris, lisez le livre de Turing cité au paragraphe 2.b.)

4. Introduction à la cryptanalyse

4.1. Les niveaux d'attaque

On fait l'hypothèse du principe de Kerckhoff, c'est-à-dire que l'attaquant O connaît le système cryptographique utilisé. Il s'agit de trouver la clef qui a servi à chiffrer le message.

La cryptanalyse d'un système cryptographique donné peut-être :

(1) Attaque à messages chiffrés connus (ciphertext-only) : O possède un ou plusieurs messages chiffrés.

(2) Attaque à messages clairs connus (known-plaintext) : O possède un ou plusieurs messages clairs avec les messages chiffrés correspondants.

(3) Attaque à messages clairs choisis (chosen-plaintext) : O a la possibilité d'obtenir la version chiffrée de messages clairs de son choix.

(4) Attaque à messages chiffrés choisis (chosen-ciphertext) : O a la possibilité momentanée de déchiffrer les messages de son choix (cette attaque concerne principalement les systèmes à clef publique).

4.2. Statistiques sur le langage

Les systèmes cryptographiques tels le chiffrement affine, le chiffrement par substitution et le chiffrement de Vigenère sont attaquables en examinant les fréquences d'apparition de chaque lettre. En comparant avec le tableau des fréquences théoriques ci-dessous, on peut retrouver, dans un message assez long, suffisamment d'indications pour le déchiffrer. On peut aussi pointer l'apparition des diphtongues et des triphtongues (groupes de 2 et 3 lettres consécutives).

Voici la table des fréquences de gauche à droite, en anglais et en français :

a :	8.20	8.25	n :	6.70	7.25
b :	1.50	1.25	o :	7.50	5.75
c :	2.80	3.25	p :	1.90	3.75
d :	4.30	3.75	q :	0.10	1.25
e :	12.7	17.75	r :	6.00	7.25
f :	2.20	1.25	s :	6.30	8.25
g :	2.00	1.25	t :	9.10	7.25
h :	6.10	1.25	u :	2.80	6.25
i :	7.00	7.25	v :	0.10	1.75
j :	0.20	0.75	w :	2.30	0.00
k :	0.80	0.00	x :	0.10	0.00
l :	4.00	5.75	y :	2.00	0.75
m :	2.40	3.25	z :	0.10	0.00

Exercice 8 : En analysant la fréquence d'apparition des lettres, déchiffrer le texte suivant, chiffré avec le système de chiffrement affine :

FMXVEDKAPHFERBNDKRXRSREFMORUD
SDKDVSHVUFEDKAPRKDLYEVLRHHRH

Le chiffrement de Vigenère dépend de m. La première étape de l'attaque est donc de calculer m. Il y a pour cela deux techniques utilisées : le test de Kasiski et l'index de coïncidence, dont on donne un aperçu ci-après.

Définition. *On appelle indice de coïncidence d'un message x de longueur n 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*

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

Si chaque lettre de l'alphabet apparaissait avec la même fréquence dans un message, alors son indice de coïncidence serait proche de $1/26 \simeq 0.0385$. En fait, les indices de coïncidence sont 0.066 pour l'anglais et 0.076 pour le français.

Ceci est utilisé de la façon suivante pour trouver la longueur m de la clef : Kasiski a remarqué que si une séquence (d'au moins trois lettres) se retrouve plusieurs fois dans le texte chiffré, alors elle provient sans doute d'une répétition dans le texte en clair, et le décalage entre les deux séquences identiques est un multiple de la longueur de la clef. On repère plusieurs coïncidences et on calcule le pgcd des décalages correspondants. C'est en général la longueur de la clef ; soit m' cette valeur estimée.

Pour le confirmer, on considère les m' sous-messages c_i pour $i = 0, \ldots, m'-1$ obtenus en prenant les caractères de rang congru à i modulo m'. Ces sous-messages doivent avoir un indice de coïncidence proche de la valeur standard, ce qui – si c'est le cas – renforce la validité de $m = m'$.

5. Introduction à la théorie de Shannon

5.1. Systèmes à sécurité parfaite

En 1949, Shannon a introduit dans la cryptanalyse la théorie des probabilités. Soit $\mathcal{S} = (\mathcal{P}, \mathcal{C}, \mathcal{K}, \mathcal{E}, \mathcal{D})$ un système cryptographique dont nous supposons que l'espace des messages en clair $\mathcal{P}$ est un espace probabilisé ainsi que l'espace des clefs $\mathcal{K}$; nous noterons $P_{\mathcal{P}}$ la mesure de probabilité sur $\mathcal{P}$, et $P_{\mathcal{K}}$ celle sur $\mathcal{K}$. On a alors le résultat suivant :

Proposition 1. *La fonction de $\mathcal{C}$ dans $\mathbb{R}$ définie par*

$$P_{\mathcal{C}}(y) = \sum_{\{K \in \mathcal{K}; y \in \mathrm{Im}(e_K)\}} P_{\mathcal{K}}(K)\, P_{\mathcal{P}}(d_K(y))$$

est une mesure de probabilité sur l'espace des messages chiffrés $\mathcal{C}$.

Pour chaque $y \in \mathcal{C}$, la quantité $P_{\mathcal{C}}(y)$ mesure la probabilité d'obtenir le message chiffré y.

Exemple : $\mathcal{P} = \{a, b\}$, $\mathcal{C} = \{1, 2, 3, 4\}$ et $\mathcal{K} = \{k_1, k_2, k_3\}$. Les mesures de probabilité étant $P_{\mathcal{P}}(a) = 1/4$, $P_{\mathcal{P}}(b) = 3/4$, $P_{\mathcal{K}}(k_1) = 1/2$ et $P_{\mathcal{K}}(k_2) = P_{\mathcal{K}}(k_3) = 1/4$, on obtient $P_{\mathcal{C}}(1) = 1/8$, $P_{\mathcal{C}}(2) = 7/16$, $P_{\mathcal{C}}(3) = 1/4$ et $P_{\mathcal{C}}(4) = 3/16$.

On remarque que $P_{\mathcal{P}}(a|3) = P_{\mathcal{P}}(a)$ et que $P_{\mathcal{P}}(b|3) = P_{\mathcal{P}}(b)$; autrement dit, le message chiffré "3" n'apporte aucune information sur le message en clair.

Rappelons que la probabilité conditionnelle $P(E_1|E_2)$ est définie par $P(E_1 \cap E_2)/P(E_2)$ si $P(E_2) > 0$, et que le théorème de Bayes dit que si $P(E_1) > 0$ et $P(E_2) > 0$, on a $P(E_2)P(E_1|E_2) = P(E_1)P(E_2|E_1)$; enfin, deux événements sont indépendants si $P(E_1 \cap E_2) = P(E_1)P(E_2)$, ce qui revient à dire que $P(E_1|E_2) = P(E_1)$.

Définition 2. *Un système cryptographique $\mathcal{S} = (\mathcal{P}, \mathcal{C}, \mathcal{K}, \mathcal{E}, \mathcal{D})$ est dit à sécurité parfaite (perfect secrecy) si pour tout $x \in \mathcal{P}$ et $y \in \mathcal{C}$ on a $P_{\mathcal{P}}(x|y) = P_{\mathcal{P}}(x)$, autrement dit, les événemnts associés à x et y sont indépendants.*

Nous laissons au lecteur le soin de faire la démonstration (facile) des résultats ci-dessous.

Lemme 3. *Soit $\mathcal{S} = (\mathcal{P}, \mathcal{C}, \mathcal{K}, \mathcal{E}, \mathcal{D})$ un système cryptographique à sécurité parfaite et tel que pour tout $y \in \mathcal{C}$, on a $P_C(y) > 0$. Alors, on a aussi $|\mathcal{K}| \geq |\mathcal{C}| \geq |\mathcal{P}|$.*

Théorème 4. *Soit $\mathcal{S} = (\mathcal{P}, \mathcal{C}, \mathcal{K}, \mathcal{E}, \mathcal{D})$ un système cryptographique vérifiant $|\mathcal{K}| = |\mathcal{C}| = |\mathcal{P}|$ ainsi que $P_C(y) > 0$ pour tout $y \in \mathcal{C}$. Alors, il est à sécurité parfaite si et seulement si toutes les clefs sont équiprobables et si pour chaque couple $(x, y) \in \mathcal{P} \times \mathcal{C}$ il existe une clef unique $K \in \mathcal{K}$ telle que $e_K(x) = y$.*

Exercice 9 : Montrer que les systèmes cryptographiques par décalage ainsi que le one time pad sont à sécurité parfaite lorsque les clefs sont équiprobables, quelle que soit la probabilité dont $\mathcal{P}$ est équipé.

5.2. Théorie de l'information

Considérons les trois expériences suivantes :

(1) On lance une pièce de monnaie. Le résultat du lancer (pile ou face) peut être codé sur un bit 0 ou 1. On dit que l'information contenue dans un lancer est de 1 bit.

(2) On lance un dé tétraédrique dont les 4 faces sont codées 00, 01, 10 et 11. Il faut $2n$ bits pour représenter n lancers.

(3) Modifions le dé ci-dessus en codant ses faces au moyen des séquences 0, 10, 110 et 111. Puisqu'aucun des codes choisis n'est préfixe d'un autre, il n'y a aucune ambiguïté pour lire une séquence de lancers à partir de son codage. Supposons que les probabilités de chaque face soit respectivement 1/2, 1/4, 1/8 et 1/8. En moyenne, pour coder le résultat d'un lancer il faut $\frac{1}{2}.1 + \frac{1}{4}.2 + \frac{1}{8}.3 + \frac{1}{8}.3 = \frac{7}{4}$ bit, soit une valeur strictement inférieure à 2.

Dans l'expérience (3), on a codé le résultat d'un lancer en utilisant une suite de bits de longueur proportionnelle à $-\log p$, où p est la probabilité du résultat. On montre que cette façon de faire est optimale si l'on veut que le nombre de bits moyen par lancer soit minimum. Ce nombre moyen s'appelle l'entropie de l'expérience et représente l'information moyenne contenue dans un résultat.

Définition 5. *Soit X une variable aléatoire discrète de loi de probabilité* $\text{Prob}\{X = x_i\} = p_i$ *pour $1 \leq i \leq n$. On appelle entropie de X la quantité*

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

La définition dépend linéairement de la base du logarithme utilisé. Si le codage est fait en bits, on utilise la base 2.

Proposition 6. *L'entropie possède les propriétés suivantes :*

(i) $H(X) \geq 0$.

(ii) Si pour tout i, $p_i > 0$, *alors on a* $H(X) \leq \log n$.

(iii) On a $H(X) = \log n$ *si et seulement si la variable* X *est équidistribuée, c'est-à-dire* $p_i = 1/n$ *pour tout* i.

(iv) $H(X) = 0$ *si et seulement si tous les* p_i *sont nuls sauf l'un d'entre eux valant 1.*

(v) On a $H(X,Y) \leq H(X) + H(Y)$ *(inégalité de Jensen).*

(vi) On a $H(X+Y) = H(X) + H(Y)$ *si et seulement si* X *et* Y *sont indépendantes.*

Pour un système cryptographique, on calcule l'entropie des variables aléatoires $\mathbb{P}$, $\mathbb{C}$ et $\mathbb{K}$ représentants respectivement le choix d'un message en clair, d'un message chiffré et d'une clef dans les espaces $\mathcal{P}$, $\mathcal{C}$ et $\mathcal{K}$.

Proposition 7. *Soient* X *et* Y *deux variables aléatoires de valeurs respectives* x_i *et* y_j *avec les probabilités* p_i *et* q_j *pour* $1 \leq i \leq m$ *et* $1 \leq j \leq n$. *L'entropie conditionnelle de* X *par rapport à* Y *est*

$$\begin{aligned} H(X|Y) &= \sum_{j=1}^{n} p_j \, H(X|y_j), \\ &= - \sum_{\substack{1 \leq i \leq m \\ 1 \leq j \leq n}} P(x_i, y_j) \log P(x_i|y_j), \\ &= - \sum_{\substack{1 \leq i \leq m \\ 1 \leq j \leq n}} q_j \, P(x_i|y_j) \log P(x_i|y_j), \end{aligned}$$

où $H(X|y_j)$ *est défini par*

$$H(X|y_j) = - \sum_{i=1}^{m} P(x_i|y_j) \log P(x_i|y_j).$$

Cette quantité mesure la moyenne de l'information sur X révélée par la connaissance de Y. On a alors les propriétés suivantes :

Proposition 8. *Si* X *et* Y *sont deux variables aléatoires, on a*

$$\begin{aligned} H(X,Y) &= H(X|Y) + H(Y), \\ H(X|Y) &\leq H(X). \end{aligned}$$

Appliquons ces résultats à un système cryptographique $\mathcal{S} = (\mathcal{P}, \mathcal{C}, \mathcal{K}, \mathcal{E}, \mathcal{D})$. On obtient :

Théorème 9. *Soit $\mathcal{S} = (\mathcal{P}, \mathcal{C}, \mathcal{K}, \mathcal{E}, \mathcal{D})$ un système cryptographique. On a alors*

$$H(\mathbb{K}|\mathbb{C}) = H(\mathbb{K}) + H(\mathbb{P}) - H(\mathbb{C}).$$

La quantité $H(\mathbb{K}|\mathbb{C})$ mesure l'information moyenne sur le choix de la clef révélée par le texte chiffré.

5.3. Entropie du langage

Dans une suite au hasard de lettres prises dans l'alphabet usuel, l'information portée par chaque lettre est $\log_2(26) \simeq 4.7$ bits. En tenant compte de la fréquence des différentes lettres, on obtient $H(\mathbb{Z}/26\mathbb{Z}) \simeq 4.19$ bits en anglais et $H(\mathbb{Z}/26\mathbb{Z}) \simeq 4.14$ bits en français. Si on considère la fréquence des diphtongues, on trouve $H((\mathbb{Z}/26\mathbb{Z})^2) \simeq 7.8$ bits, soit seulement 3.9 bits par lettre. En tenant compte des blocs de n lettres, on est amené à la définition de l'entropie d'une langue.

Définition 10. *Soit L un langage sur un alphabet $\mathcal{A}$. Son entropie est définie par*

$$H(L) = \lim_{n\to\infty} \frac{1}{n} H(\mathcal{A}^n).$$

La redondance de L est définie par

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

L'entropie de la langue anglaise, calculée par des statisticiens et des linguistes, est d'environ 1.25 bit. En moyenne, une lettre d'un texte anglais ne porte donc guère plus d'un bit d'information. Cette propriété est exploitée par les algorithmes de compression de fichiers. Un fichier compressé est tel que l'information moyenne portée par chaque bit est 1. Il est donc beaucoup plus court tout en portant la même information. La redondance de la langue anglaise est $R(L) \simeq 0.75$.

Exercice 10 : Quelle quantité d'information porte une plaque minéralogique avec 4 chiffres, 2 lettres et 2 chiffres ?

Exemple : Les fausses clefs.

Soit $\mathcal{S} = (\mathcal{P}, \mathcal{C}, \mathcal{K}, \mathcal{E}, \mathcal{D})$ un système cryptographique. Soient $x = x_1x_2\ldots x_n$ un message en clair et $y = y_1y_2\ldots y_n$ le message chiffré correspondant. On veut trouver la clef utilisée.

O attaque connaissant y : le résultat de son attaque est de trouver un certain nombre de clefs possibles.

Exemple : "WNAJW" codé avec le chiffrement par décalage ; cela peut être "RIVER" ou "ARENA" correspondants aux deux clefs 5 : $R \mapsto W$ et 22 : $A \mapsto W$. Une seule est pourtant correcte. L'autre est une fausse clef.

On peut donner une borne pour le nombre de fausses clefs.

Proposition 11. *Si $|\mathcal{C}| = |\mathcal{P}|$ et si les clefs sont équiprobables, alors, étant donné un message chiffré de longueur n ($n >> 1$), le nombre de fausses clefs s_n est tel que*

$$s_n \geq \frac{|\mathcal{K}|}{|\mathcal{P}|^{nR(L)}} - 1,$$

où $R(L)$ est la redondance du langage utilisé.

Ceci permet de définir la distance d'unicité d'un système cryptographique, c'est-à-dire la valeur de n à partir de laquelle le nombre de fausses clefs devient nul, c'est-à-dire la longueur n_0 minimale d'un texte chiffré nécessaire pour pouvoir déterminer de façon unique la clef, sous réserve de disposer d'un temps de calcul infini. On a :

$$n_0 \simeq \frac{\log |\mathcal{K}|}{R(L) \log |\mathcal{P}|}.$$

Par exemple, pour le chiffrement par substitution, avec $R(L) = 0.75$, on trouve $n_0 = 25$, ce qui veut dire qu'il suffit de disposer d'un texte chiffré de 25 lettres pour trouver la clef.

6. Data Encryption Standard (DES)

Cela fût le système cryptographique le plus utilisé dans le monde, publié en 1975 et adopté par le National Bureau of Standards en 1977. Réévalué régulièrement et renouvelé, il s'est éteint en 1998, au profit de RSA.

6.1. Principe

DES chiffre un message en clair de 64 bits en utilisant une clef K de 56 bits afin de donner un message chiffré de 64 bits. Le chiffrement se fait en 3 étapes.

Etape 1. Etant donné le message en clair x de 64 bits, on construit une chaîne x_0 en permutant les bits de x selon une permutation initiale (Initial Permutation) fixée IP : $x_0 = IP(x) = L_0R_0$, où L_0 sont les 32 bits de gauche (left) de x_0 et R_0 sont les 32 bits de droite (right).

Etape 2. On calcule ensuite 16 itérations d'une certaine fonction f, donnant successivement L_iR_i ($1 \leq i \leq 16$) suivant les règles suivantes :

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

où $\oplus$ désigne le "ou" exclusif entre deux chaînes de bits ($1 \oplus 0 = 1$, $1 \oplus 1 = 0$, $0 \oplus 1 = 1$ et $0 \oplus 0 = 0$; L_i désigne les 32 bits de gauche et R_i les 32 bits de droite d'un mot de 64 bits).

La fonction f (voir après) et $K_1, K_2, \ldots, \mathbb{K}_{16}$ sont des chaînes de 48 bits calculées en fonction de la clef K de départ (en fait, K_i est une permutation d'une sélection de certains bits de K). La suite $K_1, K_2, \ldots, K_{16}$ est le schéma des clefs (key schedule). La fonction f a deux arguments : le premier est une chaîne de longueur 32, et le second est de longueur 48 ; elle produit une chaîne de longueur 32 ; voir les détails ci-dessous.

Etape 3. On applique la permutation inverse IP^{-1} à la chaîne $R_{16}L_{16}$, pour obtenir le message chiffré y (noter l'ordre inversé).

6.2. La fonction f

Elle opère en 4 étapes sur ses deux arguments.

Etape 1. Le premier argument A de longueur 32 est expansé en une chaîne de longueur 48 suivant une fonction d'expansion (Expansion function) E. $E(A)$ consiste en les 32 bits de A permutés avec 16 bits répétés une fois.

Etape 2. Calculer $E(A) \oplus J$, où J est le second argument (de longueur 48) et écrire le résultat comme la concaténation de 8 chaînes de 6 bits
$B = B_1B_2B_3B_4B_5B_6B_7B_8$.

Etape 3. On utilise 8 S-boîtes $S_1 \ldots S_8$. Chaque S_i est un tableau fixé de 4 lignes et 16 colonnes contenant des nombres entre 0 et 15. Etant donnée une chaîne de longueur 6, par exemple $B_j = b_1b_2b_3b_4b_5b_6$, on calcule $S_j(B_j)$ de la manière suivante : les deux bits b_1b_6 détermine en base deux le numéro r de la ligne ($0 \leq r \leq 3$), et les 4 bits $b_2b_3b_4b_5$ déterminent l'écriture binaire du numéro de colonne c de S_j ($0 \leq c \leq 15$). Alors, $S_j(B_j)$ est défini par le coefficient $S_j(r, c)$ écrit en binaire comme une chaîne de longueur 4. On calcule ainsi $C_j = S_j(B_j)$ pour $1 \leq j \leq 8$.

Etape 4. La chaîne $C = C_1C_2C_3C_4C_5C_6C_7C_8$ de longueur 32 est permutée suivant une permutation fixée P. Le résultat est une chaîne $P(C) = f(A, J)$.

Il reste à donner les détails sur la permutation initiale IP, le calcul des K_i pour $i = 1$ à 16, la fonction d'expansion E, et les 8 S-boîtes. On trouvera tous les détails dans les pages de l'annexe de cette partie, ainsi qu'un exemple de chiffrement par DES.

6.3. La controverse DES

(1) Tous les calculs de chiffrement par DES sont de complexité linéaire, excepté les S-boîtes qui sont donc vitales pour la sécurité du système cryptographique DES.

(2) Or, les choix des S-boîtes ne sont pas complètement clairs. Certains ont affirmé que ces S-boîtes contenaient des pièges cachés permettant à la NSA (National

Security Agency, bien plus secret que le FBI ou la CIA) de déchiffrer tous les messages chiffrés avec DES. Cela n'a jamais pu être prouvé.

(3) Il y a $2^{56} = 72\ 057\ 594\ 037\ 927\ 936$ clefs possibles, ce qui est beaucoup et trop peu à la fois.

(4) En 1977, Diffie et Hellman affirment que l'on peut construire un VLSI chip (very large scale integrated chip) pouvant tester un million de clefs par seconde, soit les essayer toutes en un jour, le tout pour 20 millions de dollars.

(5) En 1993, M. Wiener donne la description d'une machine pour trouver la clef pour un prix de 1 million de dollars ; la machine teste 50 millions de clefs par seconde.

(6) La méthode de cryptanalyse dite différentielle de Biham et Shamir a permis d'attaquer avec succès une version de DES limitée à 8 rounds (calcul de f) au lieu des 16 rounds. Mais il semble pourtant que les jours de DES sont comptés ...

6.4. Les avantages de DES

Le système DES est très efficace : il n'y a que des permutations et des ou exclusifs entre chaînes. En 1992, DEC annonce la fabrication d'une puce de 50 000 transistors qui peut chiffrer 1 Gigabit par seconde à 250 Mhz pour 300 $. En 1991, il y avait 45 firmes ayant construit une machine à chiffrer validée par le National Bureau of Standards. DES était alors utilisé par les banques pour leurs transactions financières, par le Département de l'Energie, le Ministère de la Justice (américains) et la Banque Fédérale.

En France, il est interdit d'utiliser DES sans avoir demandé une autorisation au Ministère de l'Intérieur ; la longueur de la clef est limitée, permettant à un expert d'attaquer le système, même pour des communications privées. La législation devrait en principe être assouplie et la longueur des clefs augmentées.

De toute façon, le système RSA a supplanté DES, malgré la plus grande complexité des opérations qui réduisent d'environ 1500 fois la vitesse de chiffrement.

Enfin, on utilise DES sous 4 formes : ECB, CBC, CFB et OFB, les deux dernieres étant des chiffrements en chaîne synchrone (stream cipher) pour les communications par satellite.

Aujourd'hui, DES est toujours utilisé sur le Web pour crypter les communications. En fait, on échange des clefs par le système RSA, puis le chiffrement avec ces clefs est fait par DES, ce qui ne grêve pas la vitesse de chiffrement.

7. Le système cryptographique RSA

7.1. Introduction aux systèmes cryptographique à clef publique

Le principe des systèmes cryptographiques vus précédemment est que la règle de déchiffrement se déduit aisément de la règle de chiffrement. La clef doit donc être échangée au moyen d'un canal sûr, ou bien en utilisant un protocole spécifique.

Dans les systèmes cryptographiques que nous allons voir maintenant, il en est tout autrement. La règle de chiffrement sera largement diffusée. Le destinataire B choisit sa règle de déchiffrement et diffuse à tous ceux susceptibles de communiquer avec lui, donc A, la règle de chiffrement associée. Il lui faudra s'assurer de l'authenticité de la clef (si A veut communiquer avec B, elle doit pouvoir vérifier que la clef est bien celle de B).

Pour que ceci fonctionne, B doit connaître une information qui doit être impossible (difficile) à retrouver par quelqu'un qui ne connaît que la clef publique. La construction d'un tel système est basée sur l'existence de fonctions issues des mathématiques ou de l'informatique dites à sens unique (one-way function) qui sont facilement calculables, mais telles que leur fonction réciproque est en pratique impossible à calculer.

B choisit donc sa clef secrète et utilise une fonction à sens unique pour calculer la clef publique qu'il diffuse. On peut aussi baser un tel système sur l'existence de fonctions trappes (trapdoor functions) dont la réciproque ne peut être calculée, sauf par quelqu'un qui connaît une information supplémentaire tenue secrète.

Un certain nombre de systèmes à clef publique ont été décrits. Citons :

(1) RSA, de R. Rivest, A. Shamir et L. Adleman, basé sur la difficulté de factoriser des grands entiers.

(2) Merkle-Hellman knapstack, basé sur le problème du sac à dos. Ce système a été prouvé non sûr, sauf dans la version de Chor-Rivest.

(3) Mc Eliece, basé sur la théorie des codes algébriques.

(4) El Gamal, basé sur le logarithme discret dans les corps finis.

(5) Chor-Rivest, basé sur le sac à dos.

(6) Courbes elliptiques, basé sur le logarithme discret sur le groupe des points entiers sur une courbe elliptique sur $\mathbb{Z}/n\mathbb{Z}$.

One-way function : c'est donc une fonction $f : X \to Y$ telle que $f(x)$ est facile à calculer pour tout $x \in X$ et telle que si $y \in \operatorname{Im} f$ il est pratiquement impossible de calculer $x \in X$ tel que $y = f(x)$ pour la plupart des éléments de $\operatorname{Im} f$.

Exemple : Soit $\mathbb{P}$ l'ensemble des nombres premiers, et $f : \mathbb{P} \times \mathbb{P} \to \mathbb{N}$ définie par $f(p_1, p_2) = p_1 p_2$.

Trapdoor one-way function : c'est une foonction one-way telle que si on connaît une information supplémentaire (trapdoor information) il est facile de trouver $x \in X$ tel que $f(x) = y$.

Exemple : Soient $p = 48611 \in \mathbb{P}$, $q = 53993 \in \mathbb{P}$ et $n = pq = 2624653723$. Soit $X = \{1, 2, 3, \ldots, n-1\}$ et $f : X \to X$ définie par $f(x) = x^3 \mod n$. Si p et q sont très grands et inconnus, il est difficile de calculer $f^{-1}(y)$. Si on connaît p et q c'est facile.

L'existence de telles fonctions est impossible à prouver rigoureusement.

7.2. Description du système RSA

Le principe est le suivant :

(1) Le destinataire B choisit deux entiers n et e tels que $n = pq$ avec p et q premiers distincts et grands, et e compris entre 0 et n tel que e soit premier avec le produit $(p-1)(q-1)$.

(2) B calcule l'entier d tel que $0 \leq d \leq n$ et $ed \equiv 1 \mod (p-1)(q-1)$. Il diffuse les entiers n et e tout en gardant secret les entiers p, q et d.

(3) Pour envoyer un message à B, A le convertit en une suite d'éléments de $\mathbb{Z}/n\mathbb{Z}$. Pour chiffrer $m \in \mathbb{Z}/n\mathbb{Z}$, elle calcule $c = m^e \mod n$ qu'elle transmet à B.

(4) B déchiffre c en calculant $c^d \mod n$.

Exercice 11 : Montrer que $c^d \equiv m \mod n$.

La fonction $e \mapsto d$ est un exemple typique de fonction trappe. Il est facile de calculer l'inverse d de e modulo $(p-1)(q-1)$, mais un éventuel attaquant ne connaissant que n et e ne pourra pas calculer d. On montre en effet que la connaisance de la valeur de $\varphi(n)$ (indicateur d'Euler de n) est équivalente à la factorisation de n et que si on réussit à calculer d connaissant n et e, alors on peut factoriser n.

On peut décrire le système RSA comme au début du cours : $\mathcal{P} = \mathcal{C} = \mathbb{Z}/n\mathbb{Z}$; $\mathcal{K} = \{(n, p, q, e, d) \mid n = pq,\ p, q \text{ premiers},\ ed \equiv 1 \mod \varphi(n)\}$.

Pour $K \in \mathcal{K}$ et $x \in \mathcal{P}$, $e_K(x) = x^e \mod n$ et $d_K(x) = x^d \mod n$.

Dans un annuaire public se trouve le couple (n, e) et le triplet (p, q, d) est tenu secret par B.

Typiquement la taille de p et q est de l'ordre de 100 chiffres décimaux chacun. Certains modules RSA utilisent 512 bits (154 chiffres décimaux) ce qui ne garantit plus la sécurité aujourd'hui. Pour des raisons techniques (voir la seconde partie du cours), il faut que $p - q$ soit grand, que les nombres $p \pm 1$ et $q \pm 1$ aient chacun un grand facteur premier disons de plus de 100 bits. Si r est le plus grand facteur premier de p (resp. q), il faut que $r - 1$ ait aussi un grand facteur premier. Et enfin, il faut que les ordres de e modulo $p - 1$ et $q - 1$ soient grands.

Exercice 12 : Supposons que $e = 3$ et qu'Alice envoie le même message à 3 destinataires distincts qui ont chacun les modules publics n_1, n_2 et n_3 et le même exposant $e = 3$. Montrer que si Oscar est en possession des 3 versions chiffrées du message, il a toutes les chances de pouvoir le déchiffrer.

Exercice 13 : Supposons que A dispose de la règle de chiffrement e_A publique et de la règle de déchiffrement d_A privée, et que B dispose de manière analogue des règles e_B et d_B. Soit x le message en clair qu'A veut envoyer à B. A envoie $y = e_B(d_A(x))$ que B déchiffre au moyen de la règle $x = e_A(d_B(y))$. Montrer qu'en même temps ce procédé permet de s'assurer que c'est bien A qui a envoyé le message à B.

L'implantation actuelle de RSA utilise un modulus (n) de 512 bits et permet de chiffrer environ 600 KO par seconde. C'est beaucoup moins rapide que DES (environ 1500 fois moins), pour une sécurité semblable. On l'utilise donc pour chiffrer des messages courts, par exemple la clef d'un DES, et on chiffre ensuite le message lui-même par DES.

8. Le cryptosystème El gamal

Ce système cryptographique à clef publique est basé sur le problème du logarithme discret.

Problème : Soit p un grand nombre premier et g une racine primitive modulo p, c'est-à-dire un générateur du groupe cyclique multiplicatif des éléments inversibles de $\mathbb{Z}/p\mathbb{Z}$ (rappelons que $\mathbb{Z}/p\mathbb{Z} = \mathbb{F}_p$ est un corps commutatif ; le groupe des inversibles est donc $\mathbb{F}_p^*$). Le problème du logarithme discret est de trouver, étant donné $A \in \mathbb{Z}$ tel que p ne divise pas A, l'entier a tel que $g^a \equiv A \mod p$, avec $0 \leq a \leq p-2$. Ce problème semble être aussi difficile que la factorisation des entiers.

Principe du système El gamal

(1) Bob choisit un grand nombre premier p ainsi qu'une racine primitive modulo p et un entier b dans l'intervalle $[1, p-2]$. Il calcule $B = g^b \mod p$. Il publie les entiers (p, g, B) et garde B secret.

(2) Pour chiffrer son message $m \in \mathbb{Z}/p\mathbb{Z}$, Alice choisit au hasard un entier k dans l'intervalle $[1, p-2]$ et calcule $K = g^k \mod p$, puis $c = mB^k \mod p$ et envoie le couple (K, c) à Bob.

(3) Bob peut déchiffrer le message en calculant $cK^{-b} \equiv m \mod p$.

9. Le problème de la signature digitale

On veut résoudre les trois problèmes suivants :

La signature. Celle que l'on appose sur une commande par le Web. Elle doit engager la responsabilité du signataire qui ne pourra pas la répudier. Elle ne doit pas pouvoir être imitée et doit pouvoir être vérifiée.

L'identification. Pour prouver à quelqu'un qu'il est celui qu'il prétend être, ou qu'il a le droit d'accès à un service, droit de se connecter à un ordinateur, droit d'utiliser une carte bancaire, etc ...

L'authentification. Pour garantir l'authenticité d'un document lors d'une transmission, sa conformité avec l'original. Il se peut que le destinataire soit seul en mesure de vérifier l'authenticité.

Définition 1. *Un procédé de signature est la donnée de*
un ensemble fini $\mathcal{P}$ appelé espace des messages,
un ensemble fini $\mathcal{S}$ appelé espace des signatures,
un ensemble fini $\mathcal{K}$ appelé espace des clefs,
pour chaque clef $K \in \mathcal{K}$ une fonction de signature $s_K : \mathcal{P} \to \mathcal{S}$ et une fonction de vérification $v_K : \mathcal{P} \times \mathcal{S} \to \{oui, non\}$ telles que $v_K(x, s_K(x)) = oui$ pour tout $x \in \mathcal{P}$.

9.1. Signature RSA

Le choix des paramètres p, q, n, e, d est le même que celui du système cryptographique RSA, mais il est fait par Alice qui diffuse sa clef publique (n, e).

Pour signer un message $m \in \mathbb{Z}/n\mathbb{Z}$ qu'elle envoie à Bob, Alice calcule $s = m^e \mod n$. Dans son envoi, elle accompagne m de sa signature s.

A la réception, Bob peut contrôler la signature d'Alice en vérifiant que $s^d \mod n = m$. S'il y a 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 clef secrète nécessaire pour calculer s.

9.2. Signature El Gamal

(1) Alice choisit un grand nombre premier p ainsi qu'une racine primitive g modulo p. Elle choisit aussi un entier a dans l'intervalle $[1, p-2]$ et calcule $A = g^a \mod p$. Elle publie les entiers (p, g, A) et garde a secret.

(2) Pour signer chaque message $m \in \mathbb{Z}/p\mathbb{Z}$, Alice choisit un entier k dans l'intervalle $[1, p-2]$ premier avec $p-1$ et calcule $K = g^k \mod p$. Elle calcule aussi $s = (m - aK)\, k^{-1} \mod (p-1)$ et dans son envoi à Bob, elle accompagne le message m de sa signature (K, s).

(3) A la réception, Bob s'assure que la signature est bien celle d'Alice en vérifiant que $1 \leq K \leq p-1$ et que $A^K K^s \equiv g^m \mod p$. En effet, on a

$$A^K K^s \equiv (g^a)^K (g^k)^{(m-aK)k^{-1}} \equiv g^{aK} g^{m-aK} \equiv g^m \mod p.$$

9.3. Digital Signature Standard (DSS)

Devenu une norme fédérale en 1994. Même sécurité que la signature El Gamal mais plus courte.

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 nombre premier de 160 bits.

(1) Alice choisit un nombre premier q de 160 bits, puis un nombre premier p de $512 + 64t$ bits $(0 \leq t \leq 8)$ tel que q divise $p - 1$. Alice choisit aussi un entier g dans l'intervalle $[1, p-1]$ d'ordre q modulo p. Enfin, elle choisit a dans l'intervalle $[1, q-1]$ et calcule $A = g^a \mod p$. Elle publie les entiers (p, q, g, A) et garde a secret.

(2) Pour signer chaque message $m \in \mathbb{Z}/q\mathbb{Z}$, Alice choisit un autre entier k dans l'intervalle $[1, q-1]$ et calcule la valeur $K = (g^k \mod p) \mod q$. Elle calcule aussi $s = (m + aK)k^{-1} \mod q$ et dans son envoi à Bob, elle fait accompagner le message m par sa signature (K, s).

(3) A la réception, Bob s'assure que la signature est bien celle d'Alice en vérifiant que $1 \leq K,\ s \leq q$ et que $(A^{Ks^{-1}} g^{ms^{-1}} \mod p) \mod q = K$, où s^{-1} désigne l'inverse de s modulo q. En effet, on a

$$A^K g^m \equiv (g^k)^s \mod p,$$

donc

$$A^{Ks^{-1}} g^{ms^{-1}} \mod p = g^k \mod p.$$

En réduisant à nouveau modulo q, on obtient la relation vérifiée par Bob.

The initial permutation IP is as follows:

IP							
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

This means that the 58th bit of x is the first bit of IP(x); the 50th bit of x is the second bit of IP(x), etc.

The inverse permutation IP^{-1} is:

IP^{-1}							
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

The expansion function E is specified by the following table:

E bit-selection table					
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

The eight S-boxes and the permutation P are now presented:

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

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

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

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

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

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

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

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

P			
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

Finally, we need to describe the computation of the key schedule from the key K. Actually, K is a bitstring of length 64, of which 56 bits comprise the key and 8 bits are parity-check bits (for error-detection). The bits in positions $8, 16, \ldots, 64$ are defined so that each byte contains an odd number of 1's. Hence, a single error can be detected within each group of 8 bits. The parity-check bits are ignored in the computation of the key schedule.

1. Given a 64-bit key K, discard the parity-check bits and permute the remaining bits of K according to a (fixed) permutation PC-1. We will write PC-1$(K) = C_0 D_0$, where C_0 comprises the first 28 bits of PC-1(K) and D_0 the last 28 bits.
2. For i ranging from 1 to 16, compute

$$C_i = LS_i(C_{i-1})$$
$$D_i = LS_i(D_{i-1}),$$

and $K_i = \text{PC-2}(C_i D_i)$. LS_i represents a cyclic shift (to the left) of either one or two positions, depending on the value of i: shift one position if $i = 1, 2, 9$ or 16, and shift two positions otherwise. PC-2 is another fixed permutation.

The key schedule computation is depicted in Figure 3.3.

The permutations PC-1 and PC-2 used in the key schedule computation are as follows:

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

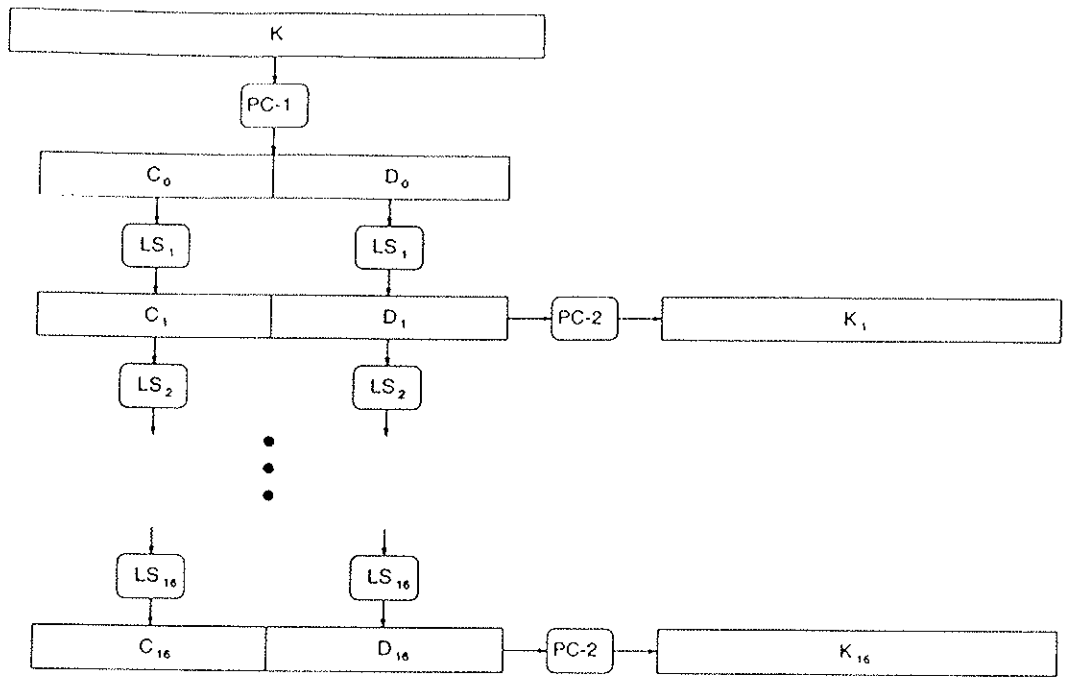

FIGURE 3.3
Computation of DES key schedule

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

We now display the resulting key schedule. As mentioned above, each round uses a 48-bit key comprised of 48 of the bits in K. The entries in the tables below refer to the bits in K that are used in the various rounds.

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

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

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

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

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

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

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

Round 8											
36	41	60	50	10	43	59	18	57	35	9	3
58	25	52	51	34	19	49	27	26	17	44	2
12	54	61	13	31	30	6	20	62	47	45	23
55	15	28	22	37	46	39	4	7	21	14	53

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

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

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

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

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

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

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

Round 16											
18	59	42	3	57	25	41	36	10	17	27	50
11	43	34	33	52	1	2	9	44	35	26	49
30	5	47	62	45	12	55	38	13	61	31	37
6	29	46	4	23	28	53	22	21	7	63	39

Decryption is done using the same algorithm as encryption, starting with y as the input, but using the key schedule $K_{16}, \ldots, K_1$ in reverse order. The output will be the plaintext x.

3.2.1 An Example of DES Encryption

Here is an example of encryption using the **DES**. Suppose we encrypt the (hexadecimal) plaintext

0123456789ABCDEF

using the (hexadecimal) key

133457799BBCDFF1.

The key, in binary, without parity-check bits, is

0001001001101001010110111100100110110111101101111111000.

Applying IP, we obtain L_0 and R_0 (in binary):

L_0	=	11001100000000000110011011111111
$L_1 = R_0$	=	11110000101010101111000010101010

The 16 rounds of encryption are then performed, as indicated.

$E(R_0)$	=	011110100001010101010101011110100001010101010101
K_1	=	000110110000000101110111111111110001110000011100 10
$E(R_0) \oplus K_1$	=	011000010001011110111010100001100110010100100111
S-box outputs		01011100100000101011010110010111
$f(R_0, K_1)$	=	00100011010010101010100110111011
$L_2 = R_1$	=	11101111010010100110010101000100

$E(R_1)$	=	011110101111010100101010000110000101010100000 01001
K_2	=	011110011010111011011001110110111100100111100101
$E(R_1) \oplus K_2$	=	000011000100010010001101111010110110001111101100
S-box outputs		11111000110100000011101010101110
$f(R_1, K_2)$	=	00111100101010111000011110100011
$L_3 = R_2$	=	11001100000000001011101110000 1001

$E(R_2)$	=	111001011000000000000010101110101110100001010011
K_3	=	010101011111111001000101001000010110011111100110 01
$E(R_2) \oplus K_3$	=	101100000011111001000100011111100000100111110010 10
S-box outputs		00100111000100000111000010110111 1
$f(R_2, K_3)$	=	01001101000101100110111010110000
$L_4 = R_3$	=	10100010010111000000101111110100

$E(R_3)$	=	01010000010000101111100000000101011111111110101001
K_4	=	011100101010110111010110110110110011010100011101
$E(R_3) \oplus K_4$	=	001000101110111100101110110111110010010101010110100
S-box outputs		00100000111101101100111110011010
$f(R_3, K_4)$	=	10111011001000110111011101001100
$L_5 = R_4$	=	01110111001000100000000001000101

$E(R_4)$	=	10111010111010010000010000000000000000001000001010
K_5	=	01111100111011000000011111101011010100111010100 0
$E(R_4) \oplus K_5$	=	11000110000001010000001111101011010100011010001 0
S-box outputs		01010000110010000011000111101011
$f(R_4, K_5)$	=	00101000000100111010110111000011
$L_6 = R_5$	=	10001010010011111010011000110111

$E(R_5)$	=	11000101010000100101111111101000011000001101011111
K_6	=	01100011101001010011111001010000011111011001011111
$E(R_5) \oplus K_6$	=	10100110111001110110000110000000101110101000000 0
S-box outputs		01000001111100110100110000111101
$f(R_5, K_6)$	=	10011110010001011100110100101100
$L_7 = R_6$	=	11101001011001111100110101101001

$E(R_6)$	=	11110101001010110000111111100101101010110101010011
K_7	=	11101100100001001011011111111011000011000101111100
$E(R_6) \oplus K_7$	=	00011001101011111011100000010011101100111111101111
S-box outputs		00010000011101010100000010101101
$f(R_6, K_7)$	=	10001100000000101000111000010011 1
$L_8 = R_7$	=	00000110010010101011101000010000

$E(R_7)$	=	00000000110000100101010101011111101000000010100000
K_8	=	11110111100010100011101011000001001110111111111011
$E(R_7) \oplus K_8$	=	11110111010010000011011111001111001111011010111011
S-box outputs		01101100000011000011111100101011110
$f(R_7, K_8)$	=	00111110000000111010000110111111001
$L_9 = R_8$	=	11010101011010010100101110010000

$E(R_8)$	=	01101010101010110101001010100101011111100101000001
K_9	=	11100000110110111111010111111011011110011111000000 1
$E(R_8) \oplus K_9$	=	10001010011100001011100101001000100110110010000 0
S-box outputs		00010001000011000101011101110111
$f(R_8, K_9)$	=	00100010001101100111110001101010
$L_{10} = R_9$	=	00100100011111001100011001111010

$E(R_9)$	=	00010000100000011111110010110000011000011111110100
K_{10}	=	10110001111100110100011110111010010001100100111 1
$E(R_9) \oplus K_{10}$	=	10100001011100001011111011011010100001011011101 1
S-box outputs		11011010000001000101001001110101
$f(R_9, K_{10})$	=	01100010101111001001110000100010
$L_{11} = R_{10}$	=	10110111110101011101011110110010

$E(R_{10})$	=	010110101111111010101011111010101111111010110100101
K_{11}	=	001000010101111111010011110111101101001110000110
$E(R_{10}) \oplus K_{11}$	=	011110111010000101111000001101000010111000100011
S-box outputs		011100110000010111010001000000001
$f(R_{10}, K_{11})$	=	111000010000010011111010000000010
$L_{12} = R_{11}$	=	110001010111100000111110001111000

$E(R_{11})$	=	011000001010101111110000000111111000001111110001
K_{12}	=	011101010111000111110101100101000110011111101001
$E(R_{11}) \oplus K_{12}$	=	000101011101101000000101100010111110010000011000
S-box outputs		011110111000101100100110001101O1
$f(R_{11}, K_{12})$	=	110000100110100011001111111101010
$L_{13} = R_{12}$	=	011101011011110100011000010110O0

$E(R_{12})$	=	001110101011111011111101010001111000000101111O000
K_{13}	=	100101111100010111010001111110101011101001000001
$E(R_{12}) \oplus K_{13}$	=	101011010111100000101011011101011011100010110001
S-box outputs		100110101101000110001011010011111
$f(R_{12}, K_{13})$	=	110111011011101100101001001000l0
$L_{14} = R_{13}$	=	000110001100001100010101010110l0

$E(R_{13})$	=	000011111000101100000011010001010101010101111O100
K_{14}	=	010111110100001110110111111100101110011100111010
$E(R_{13}) \oplus K_{14}$	=	010100000101010110110001011110000100110111001110
S-box outputs		011001000111100110011010111l0001
$f(R_{13}, K_{14})$	=	101101110011000110001110010101O1
$L_{15} = R_{14}$	=	110000101000110010010110000011O1

$E(R_{14})$	=	111000000101010001011001010010101100000001011011
K_{15}	=	101111111100100011000110100111101001111110000l010
$E(R_{14}) \oplus K_{15}$	=	010111111100010111010100011101111111111101010001
S-box outputs		101100101110100010001101001111O0
$f(R_{14}, K_{15})$	=	010110111000000100100111011011l0
$L_{16} = R_{15}$	=	010000110100001000110010001101O0

$E(R_{15})$	=	001000000110101000000010000011010010000011010l000
K_{16}	=	110010110011110110001011000011100001011111110101
$E(R_{15}) \oplus K_{16}$	=	111010110101011110001111000101000101011001011101
S-box outputs		101001111000001100100100001010O1
$f(R_{15}, K_{16})$	=	110010001100000001001111100110O0
R_{16}	=	000010100100110011011001100101O1

Finally, applying IP^{-1} to L_{16}, R_{16}, we obtain the ciphertext, which (in hexadecimal form) is:

85E813540F0AB405.

Aspects Arithmétiques de la Cryptographie

Henri Cohen

Laboratoire A2X, U.M.R. 5465 du C.N.R.S., Université Bordeaux I,
351 Cours de la Libération, 33405 Talence Cedex, FRANCE

Les deux méthodes de cryptographie à clef publique les plus utilisées actuellement, basées sur des méthodes arithmétiques, sont d'une part la méthode RSA, basée sur la difficulté de la factorisation, et d'autre part le logarithme discret sur les courbes elliptiques, basée sur la difficulté de calculer un tel logarithme discret. Notre but va être de donner un aperçu des algorithmes arithmétiques sous-jacents, avec suffisamment de détails pour que le lecteur puisse aborder une bibliographie plus spécialisée.

1 Factorisation et Primalité

Rappelons que la méthode RSA est basée sur les deux remarques suivantes : il est facile de fabriquer à la demande des "grands" nombres premiers (grand signifie ici plusieurs centaines de chiffres décimaux), et il est par contre impossible en pratique de factoriser un produit de deux tels nombres premiers. Noter que la première remarque a un sens mathématique bien précis : depuis le début des années 1980, on a découvert des algorithmes qui permettent de produire aisément des grands nombres premiers. Par contre, la deuxième remarque est de nature moins précise : en l'an 2000, les meilleures méthodes disponibles sont totalement incapables de factoriser un nombre (arbitraire) de 300 chiffres décimaux. Ceci ne signifie pas que de telles méthodes n'existent pas. Elles sont peut-être déjà connues et sont bien cachées par les services secrets qui les possèdent. Ou bien de telles méthodes seront découvertes dans les prochains mois, ou même jamais, il est impossible de le savoir.

Quoi qu'il en soit, cette impossibilité pratique de factoriser de très grands nombres a conduit à la méthode RSA qui est l'une des méthodes cryptographique les plus utilisées actuellement, en particulier pour toutes les transactions passant par Internet.

Rappelons qu'un nombre entier $p \geq 2$ est un nombre premier s'il n'est divisible par aucun autre nombre entier que 1 et lui-même. Noter que 1 n'est pas un nombre premier. La suite des nombres premiers est infinie (théorème d'Euclide, qui se démontre très facilement), et commence par 2, 3, 5, 7, 11, etc... Un nombre qui n'est ni égal à 1 ni premier est dit composé. Le résultat fondamental est que tout entier N supérieur ou égal à 2 est égal au produit de puissances de nombres premiers distincts, et ceci de manière unique à permutation près des facteurs. Le problème pratique qui se pose est de calculer cette décomposition, appelé la *factorisation* de N. En fait, on sait depuis Fermat au dix septième siècle, et nous le verrons ci-dessous, que ce problème se décompose en 4 sous-problèmes de difficultés très inégales.

1. Essayer de démontrer que N est composé.
2. Si 1. échoue, démontrer que N est premier.
3. Si 1. réussit, trouver un diviseur non trivial de N, c'est à dire différent de 1 et de N.
4. Si on trouve un tel diviseur non trivial d, trouver la décomposition complète de N en facteurs premiers en factorisant séparément d et N/d.

Ceci appelle quelques remarques. Tout d'abord, il est clair que 4. implique que le problème de la factorisation d'un nombre peut se résoudre de manière récursive. Comme le nombre d'étapes de récursivité est très petit (au plus de l'ordre de $\log N$), on peut essentiellement ne pas en tenir compte, et considérer que la factorisation d'un nombre composé est équivalent à 3., c'est à dire à la recherche d'un diviseur non trivial de N.

D'autre part, il n'est pas du tout clair à priori qu'il y ait une différence entre 1. et 2., c'est à dire entre le fait de montrer qu'un nombre est composé ou premier. Cette différence résulte du petit théorème de Fermat que nous verrons ci-dessous. Une conséquence analogue du même théorème est qu'il nous permettra très souvent de démontrer qu'un entier N est composé, mais ne nous donnera aucune indication sur les diviseurs non triviaux de N, ce qui parait à première vue paradoxal. Voyons donc tout ceci en plus grand détail.

1.1 Tests de Non Primalité

L'outil fondamental qui nous permet de tester si un nombre est composé (mais qui ne nous permet pas de montrer qu'il est premier, du moins directement), est le petit théorème de Fermat, dont l'énoncé est le suivant.

Proposition 1. *Soit p un nombre premier et a un entier non divisible par p. On a la congruence*

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

(en d'autres termes, le reste de la division de a^{p-1} par p est égal à 1).

La démontration de ce résultat est facile. Nous en donnons deux. Une première n'utilise aucune notion mathématique particulière : considérons les nombres a, $2a$, $\ldots$, $(p-1)a$ et leurs restes r_1, r_2, $\ldots$, r_{p-1} après division par p. Comme a n'est pas divisible par p, il est clair que les r_i sont distincts deux à deux, et non nuls. Il en résulte qu'à permutation près ce sont les nombres $1, 2, \ldots, p-1$. On a donc

$$a \cdot 2a \cdots (p-1)a \equiv r_1 \cdot r_2 \cdots r_{p-1} \equiv 1 \cdot 2 \cdots (p-1) \pmod{p} \ ,$$

donc

$$((p-1)!)(a^{p-1} - 1) \equiv 0 \pmod{p} \ .$$

Comme p ne divise pas $(p-1)!$ et est premier, on en déduit que $a^{p-1} \equiv 1 \pmod{p}$.

La deuxième démonstration utilise des notions de théorie des groupes, mais est beaucoup plus utile car généralisable. On dit simplement que l'ordre d'un élément divise l'ordre du groupe. Comme p est premier, le groupe $(\mathbb{Z}/p\mathbb{Z})^*$ est de cardinal $p-1$, et comme a est premier à p, la classe de a modulo p appartient à ce groupe, donc $a^{p-1} \equiv 1 \pmod{p}$.

Du petit théorème de Fermat on peut donc immédiatement déduire le test suivant : si $N \geq 3$ est un nombre impair, et si $2^{N-1} \not\equiv 1 \pmod{N}$, alors N est composé. Ceci nous conduit à trois remarques importantes.

Remarques

1. Ceci n'est *pas* une condition nécéssaire et suffisante de non primalité. Par exemple, $N = 341 = 11 \cdot 31$ vérifie $2^{N-1} \equiv 1 \pmod{N}$, et pourtant il n'est pas premier. Toutefois de telles exceptions, bien qu'en nombre infini, ne sont pas si fréquentes, donc ce test est très utile (et de plus il peut être renforcé, voir ci-dessous).
2. La condition du test peut être vérifiée de manière très efficace en utilisant les techniques classiques d'"élévation à la puissance en utilisant l'écriture en binaire de $N-1$ (par exemple, pour calculer a^{24}, on calcule a^2, a^3, $a^6 = (a^3)^2$, $a^{12} = (a^6)^2$, $a^{24} = (a^{12})^2$, tout ceci modulo N, ce qui nécessite 5 multiplications au lieu de 23 par la méthode naïve). En termes plus précis, ceci nécessite $O(\log N)$ operations sur des nombres de taille au plus égale à N^2. Ainsi, on peut tester des entiers ayant plusieurs *milliers* de chiffres décimaux. Ceci en en opposition complète avec des tests tels que $(N-1)! \equiv -1 \pmod{N}$, qui est une condition nécéssaire et suffisante de primalité (théorème de Wilson), mais qui malheureusement est totalement inutilisable car personne ne sait calculer efficacement $(N-1)! \bmod N$.
3. Comme il a déjà été dit, si le test ci-dessus nous dit que N n'est pas premier, c'est à dire qu'il est composé, nous savons donc que N possède des diviseurs non triviaux, et pourtant nous n'avons essentiellement *aucun* renseignement sur ces diviseurs. Ceci est la raison fondamentale qui fait que les tests de non primalité ou de primalité sont tellement plus efficaces que les algorithmes de factorisation, ce qui est l'aspect central de la méthode cryptographique RSA.

Puisque certains nombres composés N vérifient $2^{N-1} \equiv 1 \pmod{N}$ (ou plus généralement $a^{N-1} \equiv 1 \pmod{N}$), il est souhaitable de trouver des renforcements de ce test. L'un des plus populaires, dû à Rabin et Miller, est basé sur le résultat suivant, qui raffine le petit théorème de Fermat.

Proposition 2. *Soit N un nombre premier impair et a un entier non divisible par N. Ecrivons $N-1 = 2^s m$ avec m impair, et posons $q = a^m \bmod N$. Alors, soit $q \equiv 1 \pmod{N}$, soit il existe un entier t tel que $0 \leq t \leq s-1$ et vérifiant $q^{2^t} \equiv -1 \pmod{N}$.*

La démonstration de cette proposition, laissée au lecteur, résulte du petit théorème de Fermat et du fait que l'équation $x^2 = 1$ n'a que les solutions $x = \pm 1$ dans le corps commutatif $\mathbb{Z}/N\mathbb{Z}$.

Il reste encore certains nombres composés qui ne sont pas détectés par l'utilisation de ce résultat. Supposons que N soit composé. Si a est un entier non divisible par N qui vérifie le résultat de la proposition, nous dirons que a est un "faux témoin" de la primalité de N, puisqu'il "ment" au sujet de cette primalité. Le point essentiel du test de Miller–Rabin est que l'on peut démontrer que le nombre de faux témoins d'un nombre composé N est au plus égal à $N/4$. En d'autres termes, si on prend au hasard une valeur de a entre 2 et $N-1$, on a au plus une chance sur 4 de tomber sur un faux témoin. Le test de Miller–Rabin est alors le suivant : on choisit au hasard 20 valeurs de a entre 2 et $N-1$. Si, pour un de ces a les conditions de la proposition ne sont pas satisfaites, cela *démontre* que N n'est pas premier (comme d'habitude sans donner d'indication sur les facteurs de N). Si, par contre, les conditions de la proposition sont vérifiées pour les 20 valeurs de a, nous n'avons absolument *rien* démontré, mais nous sommes absolument (mais non mathématiquement) certains que N est effectivement premier. De fait, si les tests étaient indépendants (mais ils ne le sont pas), on pourrait dire que la probabilité que N est composé est inférieure à 4^{-20}, ce qui est négligeable.

Plusieurs auteurs ont à juste titre critiqué le test ci-dessus à cause de la non indépendance des différents essais, et ont suggéré d'utiliser plutôt une combinaison de ce test avec des tests différents, basés sur des variantes du théorème de Fermat appliqué à des groupes autres que $(\mathbb{Z}/p\mathbb{Z})^*$. Bien que ceci soit une critique tout à fait valable, par simplicité le test de Rabin-Miller demeure le test de base qui est utilisé pour se convaincre moralement qu'un nombre est premier, avant d'attaquer l'algorithme plus difficile consistant à *démontrer* mathématiquement qu'il l'est effectivement. De toutes façons, c'est sans risque puisque s'il ne l'est pas, le test de primalité le détectera.

Notons que le test de Rabin–Miller ou autres tests analogues est un test de *non primalité* et ne peut en aucun cas être appelé test de primalité puisqu'il ne démontre jamais qu'un nombre est premier. Pour ce faire, il faut utiliser les méthodes décrites dans le sous-paragraphe suivant.

1.2 Tests de Primalité

Dans ce sous-paragraphe, nous supposerons que $N \geq 3$ est un entier impair qui a réussi un certain nombre de tests de type Rabin–Miller, ce qui nous rend moralement certain que N est premier. Il s'agit maintenant de le *démontrer*. Ceci est plus difficile, mais c'est l'un des grands succès algorithmiques de ces 20 dernières années que ce problème peut maintenant être considéré comme résolu, aussi bien d'un point de vue pratique que théorique.

Une première approche, naïve, et qui est habituellement la seule connue des non spécialistes, consiste à diviser N par tous les entiers jusqu'à

sa racine carrée (la racine carrée de N suffit, puisque si N possède un diviseur d, le nombre N/d est aussi un diviseur de N, et soit d, soit N/d est plus petit ou égal à $\sqrt{N}$). Ceci répond d'ailleurs aux trois questions posées dans le premier paragraphe, c'est à dire que cela permet de factoriser N ou de montrer que N est premier. Malheureusement, le nombre d'opérations nécéssaires devient rapidement prohibitif, et il est hors de question d'utiliser une telle méthode si N a plus d'une quinzaine de chiffres décimaux.

Historiquement, les premier tests de primalité ont été trouvés au début du vingtième siècle. On a par exemple le résultat suivant, dû à Pocklington et amélioré par Lehmer :

Proposition 3. *Soit $N \geq 2$ un entier. Le nombre N est premier si et seulement si, pour tout diviseur premier p de $N-1$ on peut trouver un entier a_p tel que $a_p^{N-1} \equiv 1 \pmod{N}$ et $\mathrm{PGCD}(a_p^{(N-1)/p} - 1, N) = 1$.*

La démonstration de cette proposition n'est qu'une application facile du petit théorème de Fermat, et est laissée au lecteur.

Le paradoxe de ce résultat est que, pour démontrer la primalité de N, il est nécéssaire de factoriser $N-1$ (ou du moins de trouver les facteurs premiers de $N-1$, ce qui revient essentiellement au même), ce qui est en général beaucoup plus difficile. Toutefois, pour certains N c'est très efficace. D'autre part, au cours des années, des renforcements de ce test ont été trouvés, soit qui utilisent seulement une factorisation partielle de $N-1$, soit des combinaisons de factorisations partielles de $N-1$, $N+1$ ou même d'autres polynômes en N. Tous ces tests sont malheureusement limités par la lenteur de la factorisation.

La première avancée significative sur ce problème a eu lieu en 1979, quand L. Adleman, C. Pomerance et R. Rumely [2], suivis par H. Cohen, H. W. Lenstra et A. Lenstra [7] [8], ont montré en utilisant des généralisations du petit théorème de Fermat aux *corps cyclotomiques* et en utilisant les *sommes de Jacobi* et les *sommes de Gauss*, que l'on pouvait obtenir un test de primalité très rapide, en temps quasiment polynomial en $\log N$ (précisément en temps $O(\log N^{c \log\log\log N})$ pour une constante $c > 0$). Ceci est devenu le premier algorithme pratique qui permet de *démontrer* la primalité de nombres de centaines de chiffres décimaux en quelques minutes, ce qui était absolument impossible auparavant.

Une conséquence peut-être inattendue de ce résultat a été de montrer que des techniques provenant de théories mathématiques plus complexes que la théorie élémentaire des nombres (ici la théorie algébrique des nombres) se révèlent être des outils importants pour la résolution de problèmes algorithmiques beaucoup plus terre-à-terre. Ceci devait d'ailleurs être confirmé de manière spectaculaire dans les années qui ont suivi avec l'introduction des courbes elliptiques dans les tests de primalité, la factorisation et la cryptographie, et à nouveau la théorie algébrique des nombres dans le crible algébrique, qui est ce qui se fait de mieux dans le domaine de la factorisation (voir ci-dessous).

Le test ci-dessus (appelé APRCL du nom de ses inventeurs) a été amélioré de diverses manières, et reste l'un des deux tests de primalité utilisés actuellement. Il est impossible malheureusement d'en donner une description, même sommaire.

Un deuxième test efficace a été inventé en 1985 par O. Atkin et F. Morain [3], en utilisant cette fois-ci la théorie des courbes elliptiques sur les corps finis. A nouveau, il est difficile de donner une description de cet algorithme. On peut toutefois dire qu'en pratique cet algorithme est comparable à l'algorithme APRCL, et d'autre part qu'il possède deux avantages. Tout d'abord (et surtout), il permet de donner un *certificat de primalité* d'un nombre N. En effet, si N possède, disons, plus de 1000 chiffres décimaux, il est fort possible que l'algorithme nécéssite plusieurs jours de calcul pour prouver que N est premier. Avec l'algorithme APRCL, si on ne fait pas confiance au résultat, on est essentiellement obligé de recommencer les quelques jours de calcul pour vérifier. Par contre, avec le certificat de primalité fourni par le test d'Atkin–Morain, on peut vérifier en quelques minutes que N est effectivement premier.

Un deuxième avantage, moins important, est qu'en moyenne le test d'Atkin–Morain est polynomial. Toutefois, comme en pratique il est de vitesse comparable au test APRCL, ceci n'a qu'une valeur théorique. D'ailleurs, en 1992 L. Adleman et M. D. Huang [1] ont montré qu'il existe un test de primalité probabiliste qui est toujours polynomial, pas seulement en moyenne, et ceci en utilisant des propriétes profondes des *courbes de genre 2 sur les corps finis.*

On voit donc qu'on peut considérer le problème de la non primalité ainsi que le problème de la primalité comme résolus, aussi bien en pratique qu'en théorie. Il reste donc maintenant le problème beaucoup plus délicat de la factorisation. Avant d'aborder ce problème, il est utile de parler un peu de courbes elliptiques, puisqu'elles sont utiles à la fois en primalité (comme nous venons de le voir), en factorisation, et en cryptographie (voir ci-dessous).

1.3 Courbes Elliptiques

Soit K un corps de caractéristique différente de 2 et de 3 (on peut aussi faire l'étude dans ces cas). En pratique, il suffit de penser à $K = \mathbb{C}$, $K = \mathbb{Q}$ ou $K = \mathbb{Z}/p\mathbb{Z}$, le corps fini à p éléments. Une *courbe elliptique* E est une équation du type

$$y^2 = x^3 + ax + b$$

avec a et b dans le corps K. L'ensemble $E(K)$ des *points* de la courbe définis sur K est l'ensemble des couples $(x, y) \in K \times K$ vérifiant l'équation ci-dessus, auquel on adjoint un "point à l'infini" de coordonnées projectives $(0, 1, 0)$. Ceci est tout à fait naturel si on considère la courbe comme une courbe projective, mais le lecteur n'ayant que peu ou pas de

souvenirs de géomtrie projective n'a qu'a simplement ajouter ce point à l'infini artificiel.

La raison primordiale qui explique l'importance des courbes elliptiques dans tous les domaines où elles sont utilisées est qu'elles possèdent une structure naturelle de *groupe algébrique*, c'est à dire une structure de groupe dont les lois s'expriment simplement avec des fonctions algébriques (en fait rationnelles) des coordonnées). Cette loi de groupe est facile à décrire en termes géométriques : c'est la méthode de la sécante et de la tangente. Plus précisément, si P et Q sont deux points sur la courbe, on trace la droite passant par P et Q (la tangente à la courbe en P si les points sont confondus). Comme l'équation de la courbe est de degré 3, il est immédiat que la droite va recouper la courbe elliptique en un troisième point R (éventuellement à l'infini). On décide alors que la somme de P et de Q est, non pas le point R (ça ne marcherait pas), mais le *symétrique* de R par rapport à l'axe des x. Il est évident que la loi est interne et commutative, que le point à l'infini est élément neutre, et que tout point possède un opposé, à savoir le symétrique par rapport à l'axe des x. Ce qui est nettement plus difficile à montrer, et nous l'admettrons (il suffit de faire le calcul, bien sûr), c'est que la loi est *associative*. En résumé, l'ensemble $E(K)$ est muni d'une structure de groupe abélien. Les formules sont les suivantes. Soient $P = (x_1, y_1)$ et $Q = (x_2, y_2)$ deux points de $E(K)$, et $P + Q = (x_3, y_3)$. On a alors

$$x_3 = -x_1 - x_2 + m^2 \quad \text{et} \quad y_3 = -y_1 + m(x_1 - x_3)$$

où

$$m = \begin{cases} (y_2 - y_1)/(x_2 - x_1) & \text{si} \quad P \neq Q \\ (3x_1^2 + a)/(2y_1) & \text{si} \quad P = Q \ . \end{cases}$$

Le cas $P \neq Q$ et $x_2 = x_1$ ne peut arriver que si $Q = -P$ (donc $y_2 = -y_1$), et dans ce cas $P + Q$ est le point à l'infini.

Les courbes elliptiques sont des objets très interessants à étudier, et ont de nombreuses applications mathématiques. Par exemple, la récente démonstration par Wiles du "dernier théorème de Fermat" utilise de manière essentielle des propriétés profondes des courbes elliptiques.

Pour nos applications plus terre à terre, nous n'avons besoin de considérer que les courbes elliptiques sur les corps finis $\mathbb{Z}/p\mathbb{Z}$ pour $p > 3$. Dans ce cas, le seul résultat dont nous avons besoin, qui n'est pas très difficile à démontrer, mais toutefois quand même pas évident, est le résultat suivant, dû à H. Hasse : soit N_p le nombre de points de $E(\mathbb{Z}/p\mathbb{Z})$ (y compris le point à l'infini). On a l'inégalité

$$|N_p - (p+1)| \leq 2\sqrt{p} \ ,$$

ou en d'autres termes

$$p + 1 - 2\sqrt{p} \leq N_p \leq p + 1 + 2\sqrt{p} \ .$$

Il faut retenir deux choses de ce résultat. Tout d'abord, l'ordre de grandeur de N_p est p. D'autre part, sans rien connaître sur la courbe elliptique

E, on a une estimation de son nombre de points modulo p qui est à plus ou moins $2\sqrt{p}$ près, ce qui est peu par rapport à p. Nous verrons ci-dessous comment on se sert de ces propriétés.

2 Factorisation

2.1 Introduction

Nous considérons maintenant le problème le plus difficile : celui de la *factorisation* d'un entier N. Nous atteignons cette étape après qu'un test de non primalité nous ait démontré que N n'est pas premier. Comme nous l'avons déja mentionné, de tels tests basés sur le petit théorème de Fermat ne donnent en gén'eral aucune indication sur les diviseurs de N. Rappelons également que pour nous, factoriser signifiera simplement trouver un diviseur non trivial de N, puisqu'on peut ensuite factoriser complètement par récursivité.

Remarquons tout d'abord qu'il y a une différence philosophique fondamentale entre les tests de primalité et la factorisation. Dans les tests de primalité, nous devons être complètement rigoureux pour démontrer un résultat qui, après tout, n'est que du type oui/non.

Au contraire, pour la factorisation, nous pouvons utiliser toutes les méthodes que nous voulons, y compris en faisant des hypothèses mathématiques hasardeuses ou même fausses : en effet, la seule chose qui compte est de trouver un diviseur non trivial, et on peut complètement oublier comment il a été obtenu puisqu'il est immédiat de vérifier si c'est ou non un diviseur. En conséquence, de nombreuses méthodes de factorisation ont été inventées, et la plupart sont encore utilisées puisque certaines d'entre elles peuvent donner de bons résultats là où d'autres vont échouer (bien que pour des factorisations de très grands nombres, disons de 100 à 200 chiffres décimaux, seulement les deux méthodes les plus modernes sont utilisées).

Il est normal de commencer par diviser N par des petits nombres premiers (au plus jusqu'à 10^6 par exemple). Ceci enlève les petits diviseurs éventuels de N, mais est bien entendu totalement insuffisant pour factoriser N puisque c'est un algorithme qui prend en gros $O(N^{1/2})$ opérations.

De nombreux algorithmes plus rapides ont été découverts depuis une trentaine d'années, et nous allons en décrire quelques uns. Le lecteur remarquera qu'ils sont en général très différents les uns des autres.

2.2 La Méthode Rho de Pollard

Une méthode très simple mais très astucieuse est la méthode rho de J. Pollard, dont le nom réapparaitra plusieurs fois (Pollard est un ingénieur anglais non mathématicien, et pourtant il a contribué de manière fondamentale aux progrès des méthodes de factorisation de ces 25 dernières années).

Prenons un exemple. Définissons la suite x_k d'éléments de $\mathbb{Z}/N\mathbb{Z}$ par $x_0 = 0$ et $x_{k+1} = x_k^2 + 1 \bmod N$. Soit p un diviseur premier de N. La suite x_k modulo p ne peut prendre qu'un nombre fini de valeurs et est donc nécessairement périodique puisqu'elle satisfait une récurrence. Sous des hypothèses raisonnables, on peut montrer que la période de la suite x_k modulo p est de l'ordre de $p^{1/2}$. En particulier, si k est un multiple de la période qui est plus grand que la longueur de la partie non périodique, $x_{2k} - x_k$ sera divisible par p, donc le PGCD de $x_{2k} - x_k$ avec N sera un multiple de p qui divise N, et habituellement égal à p. Si on prend pour p le plus petit diviseur premier de N, qui est plus petit ou égal à $N^{1/2}$, on voit donc que l'on s'attend à ce que le PGCD de $x_{2k} - x_k$ avec N soit un diviseur non trivial de N pour un k de l'ordre de $N^{1/4}$, et c'est bien ce que l'on constate en pratique.

Cette méthode est tellement simple qu'on peut la programmer en quelques lignes seulement, et elle devient très rapidement plus rapide que la méthode de divisions successives par des nombres premiers. Notons que la valeur initiale $x_0 = 0$ n'est pas très importante. Par contre, on peut changer le polynôme $x^2 + 1$ en un autre polynôme, mais certains sont exclus, tels le polynôme x^2 ou, plus subtilement, le polynôme $x^2 - 2$.

Comme boite noire, on peut écrire ce qui suit :

Poser $x \leftarrow 1$, $y \leftarrow 2$, puis tant que le PGCD de $x - y$ avec N est égal à 1, poser $x \leftarrow x^2 + 1 \bmod N$ et $y \leftarrow (y^2 + 1)^2 + 1 \bmod N$. Soit d le PGCD (non trivial) de $x - y$ et N. Si d est un diviseur non trivial de N, on donne d et on termine. Sinon l'algorithme a échoué. On peut le recommencer en prenant un autre polynôme à la place de $x^2 + 1$.

En pratique, on peut tester la périodicité plus rapidement qu'en utilisant $x_{2k} - x_k$. De plus, au lieu de calculer des tas de PGCD de $x_{2k} - x_k$ qui seront égaux à 1, on accumule plutôt le produit modulo N d'une dizaine de telles quantités, et on fait un seul PGCD. S'il est égal à 1, les 10 facteurs sont premiers avec N, et sinon on revient en arrière d'au plus 10 pas pour découvrir le PGCD non trivial.

Remarque : la suite x_k, qui possède une prépériode puis une période, peut donc se représenter graphiquement par un dessin qui ressemble à la lettre grecque ρ, d'où le nom de la méthode.

Plusieurs autres méthodes très différentes ont été inventées dont le temps d'execution est en $N^{1/4}$. Citons la méthode du kangourou de Pollard, la méthode des pas de bébés pas de géants de Shanks appliquée au groupe de classes d'un corps quadratique imaginaire [10], ou encore la méthode SQUFOF de Shanks basée sur le développement en fraction continue d'un nombre quadratique réel [11], [6]. Parmi toutes ces méthodes, seule la méthode rho est encore utilisée car elle permet d'enlever efficacement des diviseurs premiers de N de taille raisonnable.

2.3 La Méthode $p - 1$ de Pollard

Cette méthode a ceci de particulier qu'elle trouve assez rapidement non pas les *petits* facteurs premiers de N, comme le fait la méthode des divi-

sions successives ou la méthode rho, mais elle trouve plutôt les facteurs premiers p de N tels que $p-1$ n'ait que des facteurs premiers relativement petits. En particulier, le nombre p ainsi trouvé peut être assez grand (bien sûr, plus p est grand, moins il y a de chances que $p-1$ n'ait que de petits facteurs premiers). C'est une application directe, mais cette fois-ci positive, du petit théorème de Fermat. Supposons que N possède un facteur premier p tel que dans la factorisation $p-1 = \prod q^{v_q}$ de $p-1$ en puissances de nombres premiers, on ait $q^{v_q} \leq B$ pour une certaine borne B raisonnable. Il en résulte en particulier que $p-1$ divise le PPCM des entiers de 1 à B, que nous noterons $[1, \ldots, B]$. D'après le petit théorème de Fermat, si a est premier à p on aura donc $a^{[1,\ldots,B]} \equiv 1 \pmod{p}$ et donc (pour faire complètement disparaître la dépendance en p, que nous ne connaissons bien sûr pas)

$$\mathrm{PGCD}(a^{[1,\ldots,B]} - 1, N) > 1 \ .$$

Il est très rare dans ce cas que ce PGCD soit égal à N, et donc il fournit un diviseur non trivial de N.

Il n'est pas difficile de calculer $a^{[1,\ldots,B]}$ modulo N. De plus, il existe plusieurs raffinement de cette méthode qui la rende très rapide et utile. Un raffinement important, analogue au test de primalité correspondant, est de remarquer que cette méthode est basée sur le groupe multiplicatif $(\mathbb{Z}/p\mathbb{Z})^*$ du corps fini à p éléments. Si on utilise plutôt le groupe multiplicatif du corps à p^2 éléments (qui, je le rappelle, n'est *pas* $(\mathbb{Z}/p^2\mathbb{Z})^*$), on peut obtenir une méthode analogue qui marche quand $p+1$ (au lieu de $p-1$) ne possède que des facteurs premiers raisonnablement petits.

2.4 La Méthode des Courbes Elliptiques de Lenstra

La méthode $p-1$ (ou son analogue $p+1$) est séduisante, mais elle possède le grave défaut de ne s'appliquer qu'aux facteurs premiers p de N ayant des propriétés particulières. La raison principale en est que l'on utilise implicitement deux groupes liés à p, de cardinal $p-1$ et $p+1$ respectivement, et deux groupes de sont pas suffisants.

L'idée fondamentale de la méthode des courbes elliptiques de Lenstra (ECM) est l'utilisation d'un grand nombre de groupes liés à p, à savoir les courbes elliptiques sur $\mathbb{Z}/p\mathbb{Z}$. En effet, comme nous l'avons vu ci-dessus, ce sont bien des groupes. D'autre part, et c'est l'un des aspects les plus importants, leur cardinal est toujours voisin de p, et donc en particulier pas trop grand. Enfin, il est facile de calculer sur ces groupes en utilisant les formules explicites données ci-dessus, et comme ces formules ne font pas intervenir explicitement le corps sur lequel on travaille, on peut calculer modulo N (bien que $\mathbb{Z}/N\mathbb{Z}$ ne soit pas un corps), tout en sachant qu'en réduisant modulo un facteur premier inconnu p de N celà donnera le bon résultat.

Regardons de plus près cette méthode. Le calcul principal qu'il faut effectuer pour ajouter deux points sur une courbe elliptique modulo N est le calcul de m, qui est égal à $(y_2 - y_1)/(x_2 - x_1)$ si les points sont

distincts et à $(3x_1^2 + a)/(2y_1)$ si les points sont confondus. Dans les deux cas on doit effectuer une *division* modulo N. Ceci peut se faire algorithmiquement assez simplement en utilisant une variante de l'algorithme d'Euclide qui donne les coefficients de l'identité de Bezout. En effet, si z est premier avec N, il existe u et v que l'on peut aisément calculer grâce à cette variante, tels que $uz + vN = 1$. Il est clair que u est un inverse de z modulo N, et donc une quantité telle que t/z se calculer en multipliant (modulo N) t par l'inverse u de z. Bien que l'algorithme d'Euclide soit assez rapide, on utilise en fait des techniques qui permettent d'accélérer le processus que nous venons de décrire.

Toutefois, nous pouvons avoir un ennui dans le calcul : si z (c'est à dire dans notre cas $x_2 - x_1$ ou $2y_1$) n'est pas premier avec N, il ne possède pas d'inverse modulo N et l'algorithme d'Euclide échouera. Mais paradoxalement, c'est exactement ça qu'on cherche ! En effet, si z n'est pas premier avec N, cela signifie que le PGCD de z avec N n'est pas égal à 1, et comme dans les méthodes décrites ci-dessus, il n'est presque jamais égal à N, donc c'est un diviseur non trivial de N, ce que nous cherchons.

Le détail du principe de base de la méthode est donc le suivant. On choisit un certain nombre de courbes elliptiques E_i, sur chacune un point P_i, et une borne B. On calcule ensuite $[1, \ldots, B] \cdot P_i$ sur chaque courbe (puisque la loi sur E_i est notée additivement, on écrit $[1, \ldots, B] \cdot P_i$ au lieu de $P_i^{[1,\ldots,B]}$). Si le calcul se fait sans encombre, on change de courbes et/ou on augmente la borne B. Si par contre l'un des calculs a échoué parce qu'on a trouvé un PGCD non trivial avec N, on a trouvé un diviseur non trivial de N et on a gagné.

Bien entendu, tout le problème consiste à analyser cet algorithme ainsi que ses raffinements, ce qui permet de choisir de manière optimale le nombre de courbes E_i à étudier et la borne B (le choix de P_i n'est pas très important à partir du moment où un ne choisit pas un point particulier de la courbe).

On obtient le résultat conjectural suivant : le temps d'exécution de ECM est de l'ordre de

$$\exp(\sqrt{2 \log p \log\log p}) \ ,$$

où p est le plus petit diviseur premier de N. Puisque $p \leq N^{1/2}$, si on oublie la dépendance en p cela donne un algorithme de factorisation dont le temps d'exécution est conjecturalement de l'ordre de

$$\exp(\sqrt{\log N \log\log N}) \ .$$

Remarques.

1. Bien que ce temps soit conjectural, c'est bien ce qu'on observe en pratique, et de toutes façons ça n'a pas d'importance puisque c'est uniquement le temps d'exécution qui est conjectural, et non la validité de l'algorithme. Mais même si la validité était conjecturale, ça

n'aurait toujours aucune importance puisque, comme nous l'avons déjà souligné, il est immédiat de vérifier qu'un nombre calculé par un algorithme est effectivement un diviseur non trivial de N.

2. La méthode ECM est toujours utilisée actuellement pour enlever des facteurs premiers moyennement grands (jusqu'à 20 ou 25 chiffres décimaux, totalement inaccessibles même pour la méthode rho de Pollard). Par contre, on ne peut en général pas l'utiliser pour factoriser complètement quand il y a plus d'un facteur premier de plus de 25 chiffres. Seules les méthodes MPQS et NFS peuvent être utilisées pour ces factorisations lourdes.

2.5 Les Méthodes de Crible

Ce sont les méthodes de factorisation les plus modernes et les plus efficaces. Les idées de base sont les suivantes. Tout d'abord, on remarque que si on connait deux entiers x et y tels que $x^2 \equiv y^2 \pmod{N}$ mais avec $y \not\equiv \pm x \pmod{N}$, on a trouvé un diviseur non trivial de N : en effet, le PGCD de $x - y$ et de N est un tel diviseur. Il ne peut pas être égal à N (sinon $x \equiv y \pmod{N}$), ni a 1 car sinon d'après le lemme de Gauss, N divise $x^2 - y^2 = (x - y)(x + y)$ et N premier avec $x - y$ entrainerait N divise $x + y$ donc $y \equiv -x \pmod{N}$.

Mais comment trouver de tels couples x et y ? La deuxième idée fondamentale est l'utilisation de bases de facteurs. On choisit une borne B, et d'une manière ou d'une autre on essaye de fabriquer des entiers x_i dont tous les diviseurs premiers sont inférieurs ou égaux à B. Si l'on peut trouver suffisamment de tels x_i (en pratique 10 de plus que le nombre de nombres premiers plus petits ou égaux à B suffisent), c'est un exercice facile de montrer que l'on peut fabriquer deux combinaisons multiplicatives x et y des x_i (modulo N bien sûr) qui vérifieront les conditions données ci-dessus. Il suffira pour cela de résoudre un système linéaire à coefficients dans $\mathbb{Z}/2\mathbb{Z}$, les coefficients étant les exposants dans la décomposition en facteurs premiers des x_i.

Reste ensuite à trouver de tels nombres x_i. Le troisième idée est que, si on arrive à générer des entiers qui, d'une part ont un certain nombre de petits facteurs premiers, et d'autre part qui sont relativement petits par rapport à N, il devrait être raisonnablement facile de trouver parmi ces entiers des x_i qui conviennent. Pour garantir la divisibilité des entiers par de petits nombres premiers, on utilise une méthode de crible, qui est de nature très simple, mais que je ne peux détailler ici (par exemple, si P est un polynôme à coefficients entiers, et si $P(a)$ est divisible par un nombre premier p, il en ira de même de tous les entiers $P(a + kp)$). Ceci explique le nom donné aux méthodes correspondantes.

Ces méthodes ne diffèrent en fait que par la manière dont elles génèrent des petits entiers. Je ne peux pas donner beaucoup de détail, mais juste l'idée générale. Historiquement, la première méthode de ce type a été basée sur le développement en fractions continues de nombres quadratiques. En effet, ceci fournit des x_i de taille inférieure à $\sqrt{N}$, ce qui est

déjà très bien. Cette méthode (appelée CFRAC) a été ensuite détronée par la méthode du crible quadratique MPQS de Pomerance. Ici, on considère plutôt des petites valeurs de polynômes de degré 2 fabriqués à l'aide du nombre N.

Ces deux méthodes ont un temps conjectural analogue à celui de ECM, à savoir

$$\exp(\sqrt{\log N \log\log N}) \ ,$$

mais MPQS possède un avantage et un inconvénient par rapport à ECM. L'avantage est que, les opérations de base étant excessivement rapides, MPQS est nettement plus rapide que ECM pour la factorisation de grand nombres. Le désavantage est que, si N possède un facteur premier p de taille moyenne (disons 15 à 20 chiffres décimaux), ECM le trouvera plus rapidement puisque, contrairement à MPQS, il est sensible à la taille de p.

La méthode la plus récente et la plus efficace pour des factorisations très lourdes est la méthode du crible algébrique de Pollard. Ici, au lieu de chercher des *entiers* x_i qui se factorisent bien, on cherche des *nombres algébriques* qui se factorisent bien, dans des corps de nombres intimement liés à N, habituellement de degré inférieurs à 6. Bien évidemment je donne encore moins de détail sur cette méthode, mais le point important est que, à cause de la dépendance du corps de nombres par rapport à N, on obtient un temps conjectural d'exécution de l'ordre de

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

pour une constante raisonnable $c = (64/3)^{1/3}$. Ceci est donc la méthode de factorisation la plus rapide connue en l'an 2000.

2.6 Conclusion

Pour résumer cet aperçu des méthodes de factorisation, nous pouvons grossièrement classer ces algorithmes en quatre catégories différentes.

1. Les algorithmes dont la vitesse dépend de manière essentielle sur la taille des petits diviseurs premiers de N. Outre l'exemple évident de la méthode des divisions successives, il y a la méthode rho de Pollard et la méthode des courbes elliptiques ECM de Lenstra.
2. Les algorithmes qui dépendent de manı'ere essentielle de propriétés particulières des diviseurs premiers de N, mais pas directement sur leur taille. L'exemple primordial est la méthode $p-1$ de Pollard ainsi que ses variantes telles que la méthode $p+1$.
3. Les algorithmes qui ne dépendent que peu ou pas du tout des propriétés des diviseurs premiers de N, et dont le temps d'exécution est exponentiel en $\log N$ (c'est à dire de la forme N^α pour un $\alpha > 0$. Dans cette catégorie se trouvent les algorithmes de Shanks pas de bébé et pas de géant ainsi que la méthode SQUFOF.

4. Les algorithmes qui ne dépendent que peu ou pas du tout des propriétés des diviseurs premiers de N, et dont le temps d'exécution est sousexponentiel en $\log N$, c'est à dire de la forme

$$O(\exp(c(\log N)^{\alpha}(\log\log N)^{\beta}))$$

pour certaines constantes α, β et c avec $c > 0$ et $\alpha < 1$. Des exemples de tels algorithmes sont la méthode CFRAC et la méthode du crible quadratique (MPQS) de Pomerance ($\alpha = \beta = 1/2$, $c = 1$), et la méthode du crible algébrique de Pollard ($\alpha = 1/3$, $\beta = 2/3$, $c = (64/3)^{1/3}$). Bien qu'elle figure dans la première catégorie, la méthode des courbes elliptiques ECM de Lenstra rentre aussi dans cette catégorie ($\alpha = \beta = 1/2$, $c = 1$). Noter que, bien que les constantes de ECM et de MPQS soient les mêmes, MPQS est en pratique nettement plus rapide car les opérations de base sont beaucoup plus simples que les opérations sur les courbes elliptiques. Toutefois, comme nous l'avons dit, ECM garde son intérêt comme la méthode la plus efficace pour trouver des facteurs premiers d'une vingtaine de chiffres.

Puisque les méthodes modernes sont capables de factoriser un nombre de 50 chiffres décimaux en moins d'une minute sur une station de travail, les algorithmes du troisème type, prenant un temps exponentiel, ne sont plus utilisés en pratique. Ainsi, un programme maderne de factorisation se déroule habituellement comme ceci. On commence par la méthode des divisions successives, puis par la méthode rho de Pollard, puis par la méthode ECM de Lenstra, pour essayer de trouver et donc d'enlever des facteurs premiers pas trop grands (moins de 25 chiffres). Après, on essaye une variante de MPQS, de NFS, ou les deux, dépendant de la taille de N. Presque tout le travail peut être distribué sur Internet à des milliers d'ordinateurs indépendants, mais interconnectés via internet. Seul un gros système linéaire que l'on doit résoudre à la fin doit être resolu sur un gros ordinateur unique.

Une illustration de ceci a été la factorisation de la première clé RSA de 512 bits (155 chiffres décimaux) effectuée en 1999 de cette manière par une équipe dirigée par H. te Riele (voir [12]). En d'autres termes, en utilisant un effort considérable à travers internet, il est maintenant possible de factoriser un nombre arbitraire de 155 chiffres décimaux.

Pour en savoir plus. Presque tout ce qui est brièvement expliqué ici est développé en détail dans un livre de l'auteur [6].

References

1. L. Adleman and M. Huang: Primality testing and Abelian varieties over finite fields. Lecture Notes in Math. **1512**, Springer-Verlag (1992).
2. L. Adleman, C. Pomerance and R. Rumely: On distinguishing prime numbers from composite numbers. Ann. of Math. **117** (1983), 173–206.

3. O. Atkin and F. Morain: Elliptic curves and primality proving. Math. Comp. **61** (1993), 29–68.
4. W. Bosma and M.-P. van der Hulst: Primality proving with cyclotomy. Thesis, Univ. of Amsterdam (1990).
5. D. and G. Chudnovsky: Sequences of numbers generated by addition in formal groups and new primality and factorization tests. Adv. in Appl. Math. **7** (1986), 187–237.
6. H. Cohen: A course in computational algebraic number theory (third printing). GTM **138**, Springer-Verlag (1996).
7. H. Cohen and H. W. Lenstra: Primality testing and Jacobi sums. Math. Comp. **42** (1984), 297–330.
8. H. Cohen and A. K. Lenstra: Implementation of a new primality test. Math. Comp. **48** (1987), 103–121.
9. P. Mihailescu: Recent developments in primality proving. Math. Comput. Simulation **49** (1999), 193–204.
10. D. Shanks: Class number, a theory of factorization, and genera. Proc. Symp. in Pure Maths. **20**, A.M.S., Providence, R.I. (1969), 415–440.
11. D. Shanks: The infrastructure of a real quadratic field and its applications. Proc. 1972 Number theory conference, Boulder (1972), 217–224.
12. H. te Riele et al.: Factorization of RSA 512, Internet announcement, summer 1999.

Titre :

Initiation à la cryptologie

Auteurs :

Henri Cohen, Professeur à l'Université Bordeaux 1
Michel Olivier, Professeur à l'Université Bordeaux 1

Laboratoire d'Arithmétique Algorithmique eXpérimentale

Public concerné :

Enseignants de mathématiques

Résumé :

Un survol en deux étapes de la cryptologie moderne.
1ère partie : un peu d'histoire, DES, RSA, la théorie de Shannon
2ème partie : les méthodes modernes de factorisation et de tests de primalité

Mots-clés :

Cryptologie, cryptographie, cryptanalyse
DES, RSA
Factorisation, primalité

Publication :

IREM d'Aquitaine
Site Lamartine
40, rue Lamartine

33400 Talence

ISBN : 978-2-85633-010-4
EAN : 9782856330104